NAVAL POSTGRADUATE SCHOOL
# MONTEREY, CALIFORNIA

THESIS

DATA COMPRESSION STANDARDS
AND
APPLICATIONS TO GLOBAL COMMAND AND
CONTROL SYSTEM

by

Everett S. Pratt

December, 1994

Thesis Co-Advisors: Murali Tummala
                                  Paul Moose

Approved for public release; distribution is unlimited.

19950414 120

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE DECEMBER 1994.	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE DATA COMPRESSION STANDARDS AND APPLICATIONS TO GLOBAL COMMAND AND CONTROL SYSTEM

5. FUNDING NUMBERS

6. AUTHOR(S) Everett S. Pratt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis provides a comprehensive study of data compression standards used by both the military and commercial industry. Text, still imagery, video and audio compression are discussed. Several of the military standards described appear to be lacking in both performance and technology when compared to the international standards available. International standards could support the objectives as well as the functionality of the Global Command and Control System. The international standards could also provide better long term interoperability and information exchange capabilities than the currently available military standards.

14. SUBJECT TERMS Data Compression Standards, Global Command and Control System.

15. NUMBER OF PAGES 135

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

i

DATA COMPRESSION STANDARDS
AND
APPLICATIONS TO GLOBAL COMMAND AND CONTROL SYSTEM

by

Everett S. Pratt
Lieutenant, United States Navy
B.S., Northeastern University , 1988

Submitted in partial fulfillment
of the requirements for the degrees of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
from the

NAVAL POSTGRADUATE SCHOOL
December 1994
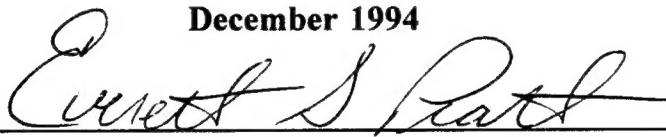
Author: _____
Everett S. Pratt
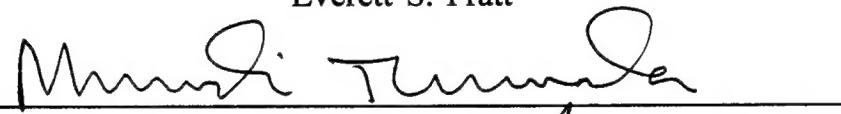
Approved by: _____
Murali Tummala, Thesis Advisor
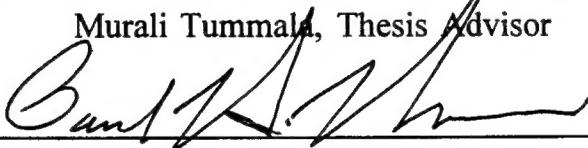
_____
Paul H. Moose, Thesis Advisor
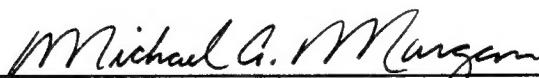
_____
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering
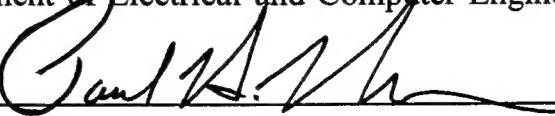
_____
Paul H. Moose, Chairman
Command Control and Communications Academic Group

iii

ABSTRACT

This thesis provides a comprehensive study of data compression standards used by both the military and commercial industry. Text, still imagery, video and audio compression are discussed. Several of the military standards described appear to be lacking in both performance and technology when compared to the international standards available. International standards could support the objectives as well as the functionality of the Global Command and Control System. The international standards could also provide better long term interoperability and information exchange capabilities than the currently available military standards.

iv

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

I. INTRODUCTION

This thesis provides a comprehensive description of data compression standards used for commercial or military applications. Due to the large number of commercial standards in use, only the newest or the most popular ones are described here. Data compression is not a new technology; it has been in use for some thirty years. The term data compression encompasses text, audio, still image and video compression. All areas of data compression are covered in the following chapters with emphasis placed on still image and video compression. The applicability of these standards to the Global Command and Control System (GCCS) is discussed in a brief but comprehensive manner.

Data compression provides a way to transmit information using fewer bits by stripping away redundancy. Early techniques of data compression were based on the information theory developed by Shannon [Ref. 1]. Every "chunk" of data contains information, and the amount of information is inversely proportional to the probability of occurrence of that "chunk". The more probable the occurrence of the "chunk" the less information it will contain. This is the basis for entropy coding which is often employed as a secondary encoding scheme in many data compression algorithms. Entropy is a measure of the amount of information contained in a set of data. It is this measure of entropy that divides data compression into two very distinct paradigms.

The first paradigm is often referred to as lossless compression. In lossless data compression no information is lost. The entropy of the compressed data set is equal to the entropy of the original source data set. The compression is achieved by reducing the average word length in the code set representing the data. Lossless compression is most often applied to text data, medical imagery and other data sources that require the decompressed data to be identical to the source data. The size of the compressed data file can be compared to the size of the original data file to obtain the compression ratio. Typically lossless compression yields approximately a 2:1 compression ratio.

Lossy data compression is the other paradigm. This method achieves much higher compression ratios than lossless compression, but at a cost to the fidelity of the resultant

1

decompressed data. Lossy compression schemes usually employ two stages of compression. The first stage is a lossy stage that results in lowering the entropy of the data set. The lossy stage is the main contributor to the higher compression ratios achieved using this method. Note that once the entropy is lowered, it can not be regained. What this means is that the information lost during lossy compression is unrecoverable. The second stage is identical to the lossless method, only it is applied to the resultant lossy compressed data set to further reduce it.

Typical applications for lossy compression algorithms are still image, video and audio signals. Many of the lossy algorithms used on these types of data use human visual and auditory perceptive limitations to determine a threshold. The threshold sets the maximum amount of compression that can be achieved without resulting in a detectable loss of information. Noticeable degradation to the decompressed data results if the threshold is exceeded. The Consultive Committee on International Telegraphy and Telephony (CCITT) uses these types of thresholds in many of the compression standards developed by their working groups. Data compression standards for military systems deal mainly with still image or video data and must adhere to the National Imagery Transmission Format Standard (NITFS).

A great need exists for a common file format that will support interoperability and data exchange independent of specific hardware and software applications. Many of the systems currently being used by the Department of Defense (DoD) and other government agencies have different file formats. These various formats often lead to problems in interoperability and incompatible file transfers. Increased requirements for images in mission planning, battle damage assessment and other military applications are magnifying these problems. Current fiscal constraints demand more efficient and economical use of existing federal assets. Redundancy and duplication of effort must be minimized within the DoD. One way to solve some of these problems is to provide a common format to share information among various federal agencies and the DoD.

The National Imagery Transmission Format Standard (NITFS) will provide a single common file format for the transmission of images and image related data between

the DoD and other federal agencies. A translator will be placed between each specific system and the transmission medium. Each translator will convert the system specific file format to the NITF prior to transmission and convert it back to the original after receipt. Figure 1 illustrates a scheme of the translation process. This simplistic scheme will help minimize preprocessing and postprocessing of image data. It will provide a complete system supporting the transmission of imagery and imagery related data. Finally, it will provide some universal functionality that is independent of the specific hardware and software being used. A list of the three primary objectives for the NITF are provided below [Ref. 2]:

1. Allow a diversity of systems to exchange imagery and imagery related data.

2. Allow a system to make a single comprehensive information transmission to a multitude of users and each user can selectively recover the respective data requested.

3. Minimize the cost and scheduling requirements to implement the NITF.

For a detailed description of the NITF refer to Military Standard 2500 (MIL-STD-2500), Chapter V. [Ref. 2]

The thesis is organized as follows. Chapter II of this thesis will focus on the compression of alphanumeric data files or text files. No official compression standard could be located for text compression; however, research indicates that both the military services and many commercial companies widely use PKZIP for text file compression. The PKZIP algorithm refers to two different coding schemes. Chapter II begins with a discussion of these two coding schemes: Lempel-Ziv-Welch (LZW) coding and Shannon-Fano coding. A detailed description of PKZIP is provided. Results of text compression
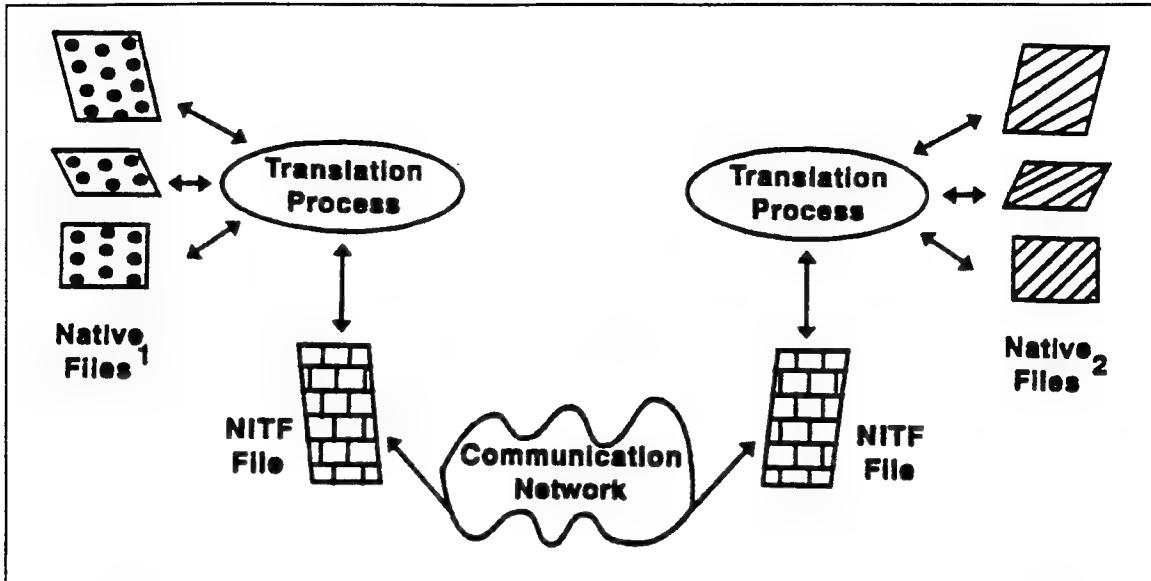
Figure 1. A graphic representation of the NITF translation process from [Ref. 2:p.14].

using Shannon-Fano coding and the PKZIP algorithm under a variety of transmission channel scenarios are included.

The main focus of Chapter III is to provide a detailed description of bi-level image compression standards. The CCITT international commercial standard or Joint Bi-Level Imagery Experts Group (JBIG) standard is discussed first. This is followed by a discussion of the current military standard used throughout the Department of Defense, MIL-STD-188-196: Bi-Level Image Compression.

The fourth chapter will focus on continuous tone still image compression standards. Like the previous chapter the first standard described is an international commercial standard or the Joint Photographic Experts Group (JPEG) standard. Military Standards 188-197 and 188-198A will be discussed in the subsequent sections.

Chapter V provides a detailed description of four different video compression standards. The CCITT recommendation H.261 video teleconferencing standard will be described first followed by the Moving Pictures Expert Group (MPEG) video compression standard for full motion video. A brief overview of MIL-STD-188-331 is provided, and

4

the chapter will conclude with an overview of Federal Information Processing Standard (FIPS) Publication 178.

The main focus of Chapter VI is audio compression standards. There are three standards that will be described in this chapter. Audio encoding for use with MPEG video encoding will be described first followed by a brief overview of the audio encoding requirements for VTC provided in MIL-STD-188-331. Chapter VI concludes with a description of MIL-STD-188-113.

A brief discussion of emerging technologies and standards is included in Chapter VII. Two developing compression techniques, namely wavelets and fractals, are discussed in this chapter. A brief overview of asynchronous transfer mode (ATM) and broadband integrated services digital network (B-ISDN) is also provided followed by an overview of the Multimedia Hypermedia Experts Group (MHEG) standard.

Chapter VIII provides a description of the functionality and objectives of the Global Command and Control System (GCCS). The chapter concludes with a discussion of the applicability of the various standards described in the previous chapters to GCCS.

Chapter IX contains the conclusions, which is then followed by three appendices. Appendix A provides an alphabetized listing of the various acronyms used throughout the thesis. Appendix B provides a listing of different standards bodies and their respective addresses and phone numbers. It also provides a listing of some of the data compression contact persons in the Department of Defense (DoD) and their phone numbers. The last appendix provides the MATLAB code that was written by the author.

II. TEXT FILE COMPRESSION

Chapter II focuses on the compression of alphanumeric data files or text files. No official compression standard could be located for text compression; however it is found that both the military services and many commercial companies use PKZIP for text file compression. The PKZIP algorithm refers to two different coding schemes. This chapter begins with a discussion of these two coding schemes. Lempel-Ziv-Welch (LZW) coding is described followed by Shannon-Fano coding. A detailed description of PKZIP is provided. Results of text compression using Shannon-Fano coding and the PKZIP algorithm under a variety of transmission channel scenarios are included.

A. LEMPEL-ZIV-WELCH CODING

Lempel-Ziv-Welch (LZW) coding does not use the probability of occurrence of symbol information like the other coding schemes described in subsequent chapters; the entropy of the data set provides the lower bound like the other methods developed. This coding scheme uses the Lempel-Ziv (LZ) algorithm with an enhancement developed by Welch. As the input data sequence is parsed into non-overlapping blocks, a dictionary table is created. This dictionary table contains each block of data and the corresponding index. If the LZ encoder transmitted an index that had not yet been added to the dictionary table in the decoder, the decoder would be unable to decode it. Welch overcame this problem by adding the "last-first" property to the LZ algorithm. The last-first property states that the symbol of the most recent word added to the dictionary table must be the first symbol of the next word added to the table [Ref. 3:p.293].

Both LZ and LZW work using the same principle which is to assign an integer index to the input strings being coded. The encoder and decoder are both initialized with identical tables by assigning an integer value to each input symbol. In the case of binary data, there are two input symbols, zero and one. The encoder simply performs a mapping from the binary symbols to the indices and transmits the indices. Using the indices and the last-first property, the decoder is able to decode the data and build the decoding

7

dictionary table. Most commercial applications are limited in the table size which limits the number of different symbols that can be encoded.

The basic LZW encoder recursively separates the input sequence into nonoverlapping blocks of varying sizes (symbols) and can be broken down into four major steps. First, the encoder looks at the remaining input sequence for the longest symbol that exists in the encoder table, and it transmits that symbol's index. In the next step, the encoder then creates a new symbol by adding the next bit in the input sequence to the current symbol and assigns this new symbol an index. A buffer containing the current input symbol being encoded is used as a pointer. The pointer then shifts in the next input symbol to be encoded (the one that has just been added to the previous symbol). The process is repeated until the entire input sequence has been encoded. An example of the encoding algorithm is provided below.

EXAMPLE 2.1: LZW ENCODER

We use the LZW algorithm to create the encoder table and show the location of the pointer at each time step for a given input sequence. In this example the encoder and decoder would have been initialized with the two binary symbols. The integer index representing each binary symbol would be identical to the symbol itself. The first bit is shifted into the pointer at time one. The respective index is transmitted, and a new symbol is created by appending the next bit to the current symbol. This new symbol is "10", and the index is two as seen in Table I below. The encoder then scans the remaining input sequence for the largest symbol, in this case a zero, and transmits the index. This cycle is repeated until the entire bit sequence has been encoded.

```
            Input sequence:  1 0 1 1 0 0 1 0 1
Pointer position at time (n):  1 2 3 4    5   6
  Encoder output sequence:      1 0 1 2 3 3
```

Time (n)	Transmitted Index	New Table Entry	New Index
1	1	10	2
2	0	01	3
3	1	11	4
4	2	100	5
5	3	010	6
6	3		

Table I. The LZW encoder table for example 2.1.

As mentioned above the decoder starts with the same initial table as that provided to the encoder. It may therefore immediately decode both a one and a zero upon receipt. After receiving these first two indices the decoder would recognize a new two bit symbol and add it to the decoding table with the appropriate index. The decoder will then decode the third index received while retaining the previous one. Using this information the decoder then creates a new symbol by appending the first bit of the current symbol to the end of the last symbol and assigning it an index. This process continues until the transmitted data stream has been completely decoded. The example below highlights this decoding process by continuing example 2.1.

EXAMPLE 2.2: LZW DECODER

Now we create the decoder table and show the location of the decoder pointer at each time step given the transmitted index sequence (from example 2.1). The decoder builds the decoding table using the output (column 3 in the table below). Index 1 is received representing a binary 1, followed by index 0 representing a binary 0. The binary output produced by these first two indices are then combined to form the first new decoder table entry 10, as shown in the table below. The new decoder table entries are

formed by appending the first binary bit of the current output sequence to the end of the previous binary output sequence.

Received sequence: 1 0 1 2 3 3

Pointer position at time (n): 1 2 3 4 5 6

Decoder output sequence: 1 0 1 1 0 0 1 0 1

Time (n)	Input	Output	Received Sequence	New Table Entry	New Index
1	1	1	1		
2	0	0	10	10	2
3	1	1	101	01	3
4	2	10	10110	11	4
5	3	01	1011001	100	5
6	3	01	101100101	010	6

Table II. The LZW decoder table for example 2.2.

B. SHANNON-FANO CODING

Shannon-Fano coding uses the entropy of the data set for a lower bound, and the upper bound contains the entropy plus a constant equal to one. The average codeword length for a Shannon-Fano code must be between these two bounds. Like Huffman coding, discussed in Chapter IV, Shannon-Fano coding uses the probability of occurrence of the symbol to determine the code word length. Each probability of occurrence must be known prior to encoding. Shannon-Fano coding is not as efficient as Huffman coding but is very close [Ref. 1]. Only when all the symbols have equal probabilities of occurrence will the average codeword length be equal to the entropy. This is also the only case when Shannon-Fano coding will produce an average codeword length that

equals the average Huffman codeword length. The Shannon-Fano codeword length is computed using [Ref. 1:p.121]

$$\log_2\left(\frac{1}{p_i}\right) \le l_i < \log_2\left(\frac{1}{p_i}\right) + 1 \qquad (1)$$

where the p_i is the symbol probability of occurrence and l_i is the codeword length. It is also important to note that Shannon-Fano codes meet the condition of a prefix code. This means that no code word is a prefix of another code word. Hamming [1] provides an example of Shannon-Fano coding, which better exhibits this characteristic.

EXAMPLE 2.3: SHANNON-FANO CODING

Given the set of symbol probabilities, we compute the respective codeword lengths, assign the codewords and produce a Shannon-Fano code tree. There are two symbols each having a probability of occurrence of 0.25 ($p_1=p_2=0.25$) and four symbols each with a probability of occurrence of 0.125 ($p_3=p_4=p_5=p_6=0.125$). The length of each respective codeword is equal to the logarithm (base two; because of binary input) of the inverse of the probability of occurrence. The two more probable symbols have a length of 2 ($l_1=l_2=\log_2(4)=2$), and the four less probable symbols have a length of 3 ($l_3=l_4=l_5=l_6=\log_2(8)=3$). The shorter codewords are assigned first beginning with the lowest number (in this case "00") and incrementing by one. The four longer codewords are created by placing a one in front of the "00" and then incrementing by one for the remaining three. The six resultant symbols would be; $s_1=00$, $s_2=01$, $s_3=100$, $s_4=101$, $s_5=110$, $s_6=111$. See Figure 2 for the Shannon-Fano tree.

Huffman coding also uses the probability of occurrence of symbols and results in a variable length code. However, unlike Shannon-Fano coding, the Huffman codeword lengths are directly proportional to the probability of occurrence. The more probable the symbol the shorter the respective Huffman codeword. For more details on Huffman coding refer to Chapter IV.
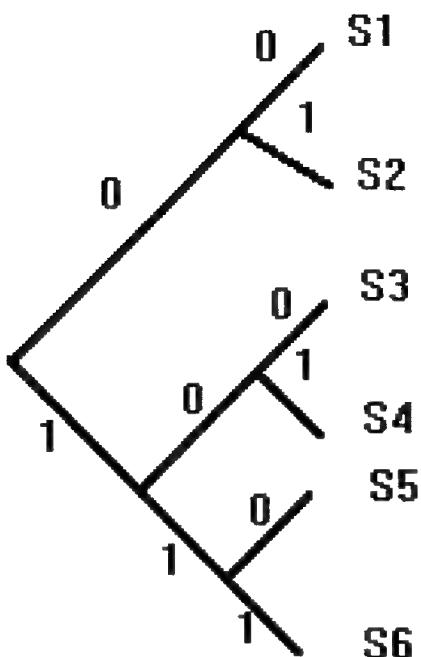
Figure 2. The Shannon-Fano code tree.

C. PKZIP

The PKZIP algorithm is widely used throughout both the commercial sector and the military. Developed by PKWARE, it is available on the Internet as shareware. This text compression algorithm has nine different modes, of which only eight are used. None of the modes used supports word alignment or bit stuffing. All of these modes can also be encrypted using the pseudo-random number generator and three keys to encrypt the files. The first mode, referred to as mode zero, provides no compression at all. Mode seven is not used by PKZIP.

Mode one is called "shrinking" and employs a dynamic LZW algorithm with partial clearing in the tables. The initial codeword size is nine bits, and the maximum is 13 bits. Shrinking works very similar to LZW with a few minor modifications. First, the codeword size is controlled by the encoder, and it is not automatically increased at the

12

decoder when a new symbol is recognized. The decoder will only add these new larger symbols if it receives a special code from the encoder to do so. Second, when the table is full, partial clearing is performed at all the leaf nodes of the LZW tree. The cleared nodes will then be reused for new symbols beginning with the lowest index. Again the encoder instructs the decoder when to perform partial clearing. [Ref. 4]

Modes two through five all use the "reducing algorithm", a combination of two compression algorithms in series. The first algorithm compresses repeated byte sequences, and the latter applies a probabilistic compression scheme to the compressed data produced from the previous one. Each ASCII character receives a "follower set" produced by the probabilistic compression scheme. Every follower set can then contain up to 32 characters. These follower sets are encoded by first encoding the size of the set and then encoding each element in the set. If a particular set was empty its length would be zero, and the decoder would automatically move on to the next one. The follower sets are stored in the data stream in reverse order at the beginning of the reduced data file followed by the compressed data. [Ref. 4]

Decompression is achieved by mapping the follower sets to each respective ASCII character. Expansion of the reduced data file (compressed data) is accomplished using one of four expansion algorithms. Each algorithm has the same three basic steps, but achieves varying degrees of compression based upon the compression factor chosen by the user.

Mode six is called "imploding", and it also uses two algorithms in series. Repeated byte sequences are compressed using a sliding dictionary, and this output is compressed further using Shannon-Fano trees. Dictionary length may be either 4k or 8k and is determined by setting a flag bit. The Shannon-Fano trees are stored at the beginning of the compressed data file. This method uses two trees in most applications and may use a third. The first tree is used to encode the length information provided by the sliding dictionary, and the second tree encodes the distance information. A third tree can be used to encode all 256 ASCII characters and is transmitted prior to the other two trees. Each Shannon-Fano tree is stored and transmitted in a compressed format. The

13

first byte of the tree represents the number of bytes minus one used to represent the Shannon-Fano tree. All the remaining bytes are used to represent the tree by employing the following scheme. The four MSBs represent the number of values at that given bit length plus one. The four LSBs determine the bit length required to represent the value plus one. After decompressing the Shannon-Fano trees, they may be used to decompress the remaining data. [Ref. 4]

Mode eight is the last one and is referred to as "deflating". It uses a 32k sliding dictionary with secondary Huffman/Shannon-Fano coding. The compressed data is stored in blocks, and a block header is used to describe the block and the Huffman coding within it. Two types of Huffman coding may be used, fixed or dynamic. Dynamic Huffman coding sends the literal, distance, and length information using 14 bits. Decompression of the Huffman codebook must be performed prior to decompressing the data stream. [Ref. 4]

D. RESULTS

This section presents the results obtained from performing data compression on four different text files. The text files were compressed using either PKZIP or a modified Shannon-Fano algorithm resulting in a total of eight compressed text files. The latter compression algorithm uses the same method described above in section II.B with one minor change. The shortest codeword length is assigned a decimal 1 and the next shortest codeword length is assigned a decimal two. This assignment process continues until the entire character set in the file has been assigned a decimal codeword. The decimal codewords are then converted into their binary equivalents resulting in variable length binary codewords. A third set of four text files, the baseline set, did not receive any type of compression. All 12 test files were then transmitted over three different transmission channels.

The first channel simulates a baseband signal using a binary symmetric channel (BSC) or a binary asymmetric channel (BAC). A block diagram of channel one is shown in Figure 3. Channel one is corrupted by uniform random noise. The probability

14

of correctly receiving a 1 given a 1 was sent, P(1|1), and the probability of correctly receiving a 0 given a 0 was sent, P(0|0), are user inputs. These two inputs can be used to define the BSC, where P(1|1)=P(0|0), or the BAC, where P(1|1)≠P(0|0). After the files are transmitted the received files are compared to the original files to determine the number of errors. Table III shows the values of P(1|1) and P(0|0) for channel one used in the analysis.
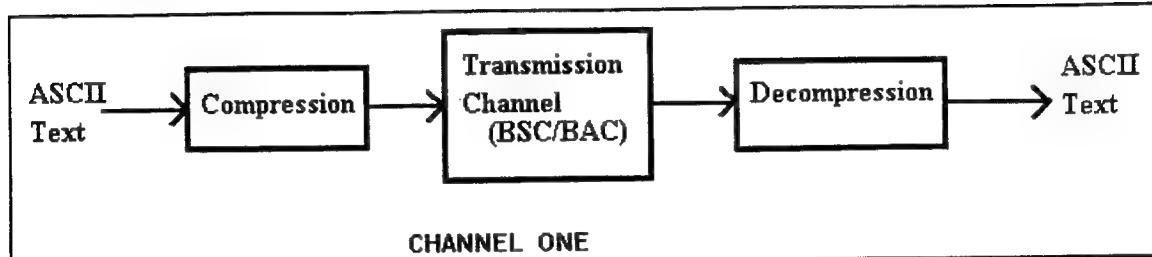
Figure 3. A block diagram of channel one.

	BSC1	BSC2	BSC3	BSC4	BAC1	BAC2	BAC3	BAC4
P(1|1)	1.0	0.995	0.99	0.98	1.0	0.995	0.995	0.99
P(0|0)	1.0	0.995	0.99	0.98	0.995	1.0	0.99	0.995

Table III. The data defining the four BSCs and four BACs used in channel one.

Channel two simulates a QPSK signal corrupted by additive white Gaussian noise (AWGN). The degree of AWGN present in the channel is determined by the standard deviation of the noise distribution, and can be controlled by the user. Channel two counts the number of bit errors by comparing the input file, at point A, to the output file, at point B, as shown in Figure 4. The error count is performed on the compressed data prior to decompression. The results from this channel simulation are shown in Table V.

The third channel also simulates a QPSK signal corrupted by AWGN. However, this channel uses a rate ½ convolutional coder to provide forward error correction (FEC). Forward error correction is used to reduce the number of errors in the resultant file by detecting and correcting them. Using FEC reduces the number of errors received, but it
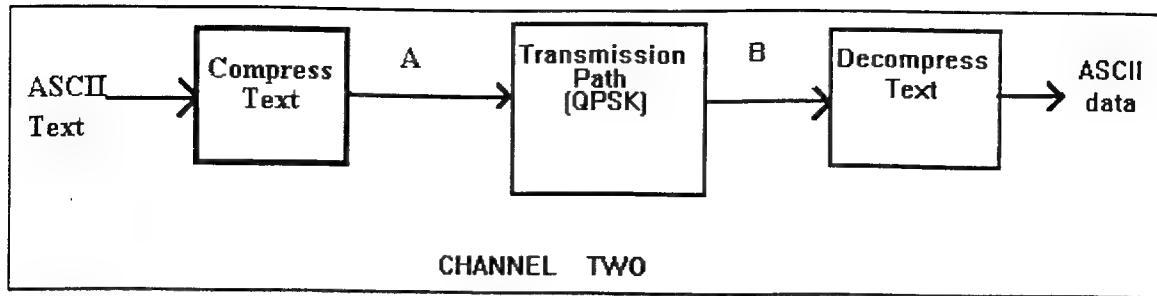
Figure 4. A block diagram of channel two.

also requires a large overhead. The rate ½ convolutional coder transmits two bits for each one that is input essentially doubling the size of the file being transmitted. If a compression scheme only averages a 2:1 compression ratio, like those used in this analysis, then using this type of FEC almost completely negates the effects of using compression. However, if FEC is already being used and text compression is added, the transmitted file would then be smaller than if it were not compressed. For more details on FEC and convolutional codes refer to Lin and Costello [Ref. 5]. Channel three also counts the number of bit errors by comparing the input file, at point A, to the output file, at point B, as shown in Figure 5. The error count is performed prior to decompression and the results are shown in Table VI. The effect of errors on each of the two types of compressed data files is described below.

The data obtained from conducting the tests is displayed in Table IV, Table V and Table VI below. Under the heading file there are three different types of entries. The first four are the plain text files labeled ONE, TWO, THREE, and FOUR, followed by the Shannon-Fano compressed files labeled SF1, SF2, SF3, and SF4 respectively. The remaining four files were compressed using PKZIP and are labeled ZIP1, ZIP2, ZIP3, and ZIP4. The number in the compressed file names is used to associate that file with the respective plain text file. The numbers in the tables represent the percentage of correctly received characters in the four plain text files and the percentage of correctly received symbols in the eight compressed text files.

The number of characters received with an error is directly related to the number bits being transmitted through the respective model. As the number of bits being

16
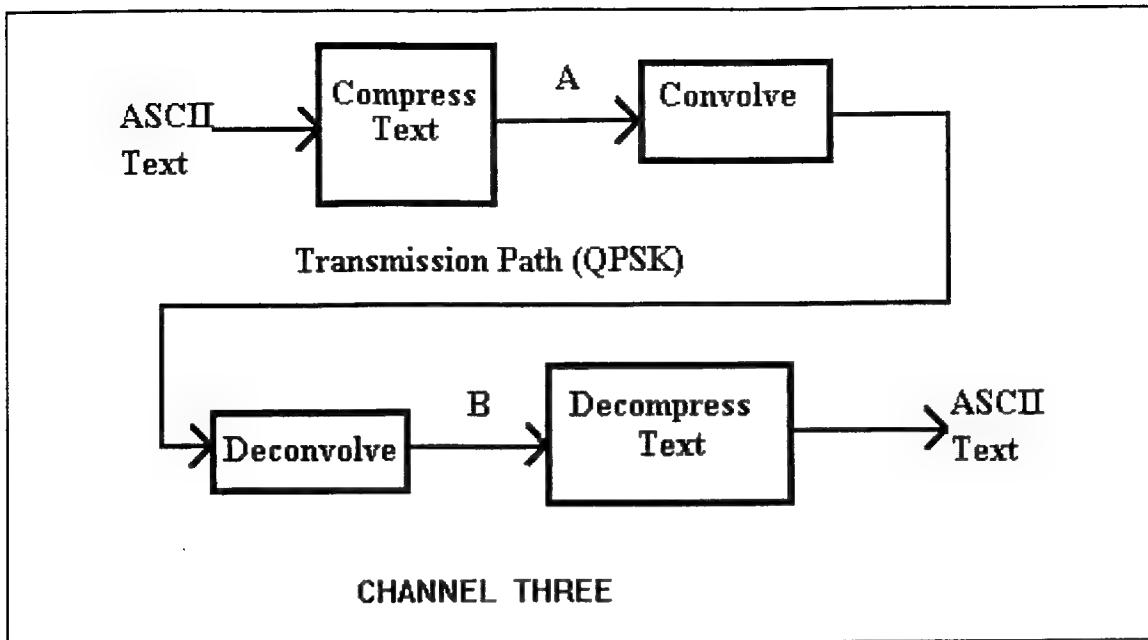
Figure 5. A block diagram of channel three using QPSK and FEC.

transmitted increases, the number of bit errors will also increase. A single character error will result from one or more bit errors in the binary representation of the character. Referring to Table IV, the files compressed using the Shannon-Fano algorithm received fewer errors than the PKZIP files and the plain text files. A text file compressed using the Shannon-Fano algorithm contains the same number of characters as the original plain text file, but the Shannon-Fano file uses fewer bits to represent the character set resulting in compression. The PKZIP algorithm achieves compression by using fewer characters to represent the original plain text file. Each character in the PKZIP file is represented by an 8-bit ASCII character. A single symbol error in a file compressed using Shannon-Fano will result in only one character error after decompression. However, a single symbol error in a file compressed using PKZIP will result in one or more character errors after decompression. These last two statements are the basis for the Shannon-Fano algorithm's outperforming PKZIP. The effect of errors after decompression is much more profound on PKZIP files than plain text or file compressed using Shannon-Fano coding. These results are included in Table IV.

17

FILE	BSC1	BSC2	BSC3	BSC4	BAC1	BAC2	BAC3	BAC4
ONE	100.0%	96.06%	92.11%	85.37%	97.93%	98.32%	94.50%	94.50%
TWO	100.0%	96.64%	92.76%	85.43%	97.64%	98.20%	94.08%	95.08%
THREE	100.0%	96.19%	92.73%	85.25%	97.05%	99.31%	93.69%	95.40%
FOUR	100.0%	94.93%	92.19%	85.67%	96.67%	97.10%	93.34%	95.07%
SF1	100.0%	98.40%	96.61%	93.62%	99.29%	99.09%	97.90%	97.56%
SF2	100.0%	98.44%	97.00%	94.04%	99.32%	98.92%	97.36%	98.08%
SF3	100.0%	99.30%	98.66%	97.26%	99.00%	99.42%	99.30%	98.20%
SF4	100.0%	98.84%	97.25%	92.76%	98.99%	98.70%	97.97%	97.11%
ZIP1	100.0%	95.97%	92.25%	85.12%	98.00%	98.48%	94.52%	94.39%
ZIP2	100.0%	96.34%	92.47%	86.71%	97.99%	98.28%	94.04%	94.11%
ZIP3	100.0%	95.93%	90.36%	87.79%	97.22%	98.29%	93.58%	93.58%
ZIP4	100.0%	96.88%	89.58%	83.75%	96.67%	96.67%	93.13%	92.71%

Table IV. The percentage of correctly received characters for the eight simulations listed using channel one.

Channel two simulates a QPSK signal being corrupted by AWGN. The four simulation runs using this channel had values of $\sigma=0.10$, 0.15, 0.20, and 0.25 respectively. The same 12 files were used in this channel, and the results are shown in Table V. The first two columns have no errors; this is due to a large signal-to-noise ratio (34dB and 27dB respectively). The last two columns have signal-to-noise ratios of 22dB and 18dB respectively, and are beginning to exhibit errors in the received files. Comparing these last two columns, we can see that all three sets of files perform equally in this channel.

The data provided in Table VI shows the effect of providing forward error correction to the three sets of files. Channel three was used in this simulation and was identical to channel two, except for the inclusion of FEC. The files compressed using the Shannon-Fano algorithm resulted in a couple errors being received, but the overall performance of both compression schemes is about equal. The inclusion of forward error correction shows a marked improvement in comparison to the results shown in Table V

FILE\σ	σ=.10	σ=.15	σ=.20	σ=.25
ONE	100.0%	100.0%	99.96%	99.63%
TWO	100.0%	100.0%	99.97%	99.66%
THREE	100.0%	100.0%	99.97%	99.70%
FOUR	100.0%	100.0%	99.98%	99.72%
SF1	100.0%	100.0%	99.97%	99.65%
SF2	100.0%	100.0%	99.99%	99.59%
SF3	100.0%	100.0^	99.93%	99.65%
SF4	100.0%	100.0%	100.0%	99.55%
ZIP1	100.0%	100.0%	99.98%	99.65%
ZIP2	100.0%	100.0%	99.98%	99.66%
ZIP3	100.0%	100.0%	99.92%	99.54%
ZIP4	100.0%	100.0%	100.0%	99.82%

Table V. The percentage of correctly received bits using simulation channel two.

for all three sets of files. The addition of FEC can reduce the adverse effects that errors can have on compressed data by correcting the errors prior to decompression. As mentioned previously, the use of FEC involves a large overhead and some of the bandwidth freed up from using compression is lost.

The final comparison to be made between the two compression schemes is the compression ratio (CR) achieved by each one. The compression ratio was calculated by dividing the size of the original plain text file by the size of the respective compressed file. These results are provided in Table VII. The column headers represent the respective plain text files, and each column lists the size of the file (in number of bits) and the compression ratio (CR) achieved. In all but one case the Shannon-Fano algorithm achieved a higher compression ratio than PKZIP. The one file where PKZIP

FILE\σ	σ=0.10	σ=0.15	σ=0.20	σ=0.25
ONE	100.0%	100.0%	100.0%	99.99%
TWO	100.0%	100.0%	100.0%	100.0%
THREE	100.0%	100.0%	100.0%	100.0%
FOUR	100.0%	100.0%	100.0%	100.0%
SF1	100.0%	100.0%	100.0%	100.0%
SF2	100.0%	100.0%	100.0%	99.99%
SF3	99.95%	99.93%	100.0%	99.98%
SF4	100.0%	100.0%	100.0%	100.0%
ZIP1	100.0%	100.0%	100.0%	99.99%
ZIP2	100.0%	100.0%	100.0%	100.0%
ZIP3	100.0%	100.0%	100.0%	100.0%
ZIP4	100.0%	100.0%	100.0%	100.0%

Table VI. The percentage of correctly received bits using channel three and the indicated values for σ. ·

outperformed Shannon-Fano contained four rows of numbers rather than text, resulting in a lot more character redundancy. The degree of redundancy found in the plain text files is limited to the natural redundancy found in the English language. The amount of compression that can be achieved on these types of files using PKZIP and Shannon-Fano also appears to be limited by this natural redundancy. Both compression schemes achieve very similar compression ratios, but the Shannon-Fano achieves slightly more compression. The slight advantage Shannon-Fano has over PKZIP appears to be due to the variable length codewords used by Shannon-Fano coding. The most frequently occurring characters in the text file receive the shortest codewords and the less frequently occurring ones receive longer codewords.

In summary, the only text file compression algorithm in wide use by both the military and the commercial industry is PKZIP. PKZIP has seven modes, each one

SET\FILE	ONE (# bits\CR)	TWO (# bits\CR)	THREE (# bits\CR)	FOUR (# bits\CR)
PLAIN TEXT	51016\none	19992\none	23328\none	5528\none
PKZIP	23232\2.20:1	11136\1.80:1	3736\6.24:1	3840\1.44:1
SHANNON-FANO	20810\2.45:1	8052\2.48:1	4236\5.51:1	2146\2.58:1

Table VII. The file size and respective compression ratio for the two compression schemes.

employing a different coding scheme or a different compression rate. The three primary coding schemes used by PKZIP are Lempel-Ziv-Welch, Shannon-Fano, and Huffman coding. Based upon the results provided in section II.D, when transmitted over noisy channels, the Shannon-Fano algorithm outperforms the PKZIP algorithm. However, these results are inclusive and further analysis should be performed on decompressed text files. Both compression schemes offer lossless text compression with at least a 2:1 compression ratio in most cases.

III. BI-LEVEL IMAGE COMPRESSION STANDARDS

The main focus of this chapter is to provide a detailed description of bi-level imagery compression standards. The CCITT international commercial standard — Joint Bi-Level Imagery Experts Group (JBIG) — is discussed first. This is followed by the current military standard used throughout the Department of Defense: MIL-STD-188-196 Bi-Level Image Compression.

A. JOINT BI-LEVEL IMAGERY EXPERTS GROUP (JBIG)

The objective of the JBIG standard is to replace the less effective CCITT Group 3 and CCITT Group 4 standards and define a single compression standard for lossless image coding [Ref. 6:p.1]. The algorithm employs lossless encoding and is considered the state-of-the-art in bi-level image compression [Ref. 7:p.1]. A predictive coding algorithm using image templates is employed by the lossless encoding scheme. Each image template is a composite of the current picture element (pixel) being encoded and previously encoded pixels to the left and above the current one.

The JBIG algorithm is also applicable to continuous tone and half-tone images. The continuous tone imagery is limited to gray scale images represented by six bits/pixel (bpp) or less. At greater bpp representations, the lossless Joint Photographic Experts Group (JPEG) algorithm would outperform JBIG [Ref. 6:p.2]. It is worth noting that in order to achieve maximum compression on a gray scale image, the bit planes should be Gray encoded before carrying out the JBIG algorithm. An example of Gray encoding is provided below.

23

EXAMPLE 3.1: GRAY ENCODING

Given the numbers one through eight, we use three bit symbols to represent each integer employing Gray encoding.

integer	Gray Code
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

Note that each symbol differs by only one bit.

1. Q-Coder

The JBIG algorithm uses a Q-coder based upon binary arithmetic coding and the estimation of the probability of occurrence of a symbol. Combining low precision probability estimation and arithmetic coding allows the coder to adapt to symbol statistics providing increased compression. The "Q" in Q-coder is used to represent the probability estimate (Q_e) of the least probable symbol (LPS). This adaption provides encoded symbols that closely match the true pixel probabilities in each bit plane. These symbol probabilities are calculated based upon the pixel probability in a specific context. [Ref. 8]

Two contexts are described in the JBIG standard, a low resolution context and a differential context. The low resolution context is the base layer containing all the low spatial frequency information. The differential context refers to all subsequent layers, and it contains the difference values between the current layer and the previous one. This coder uses the same convention as the Huffman coder i.e., the frequently occurred symbols receive shorter codewords while the seldom occurring ones receive longer codewords.

As mentioned above the Q-coder uses adaptive binary arithmetic coding and estimation of the probability of occurrence of symbols. A more detailed discussion of arithmetic coding is provided in the next chapter. One problem with adaptive arithmetic

24

coding is that it requires periodic re-estimation of the probabilities. Re-estimation of the probabilities is performed to ensure the estimates closely resemble the actual values of probability of occurrence. The estimation process of the Q-coder is implemented using a finite state machine made up of 60 states. Half of these states are for the more probable symbol (MPS) of 1, and the other 30 are for a MPS of 0. Each state has a LPS probability estimate, Q_e, associated with it. The MPS probability estimate, $P_e = 1-Q_e$. In the case of the Q-coder, this re-estimation is performed after renormalization. The value of Q_e is increased to the next higher value after LPS renormalization, and decreased to the next lower value after MPS renormalization. [Ref. 9]

Renormalization is simply a rule used to maintain the interval size within the fixed precision limits being used by the arithmetic coder and is most often implemented using a logical shift left operation [Ref. 9]. Renormalization must be performed using an identical method on both the code string and the interval size to ensure continued encoding or decoding. Both the encoder and decoder must also use the same renormalization procedure. Renormalization is performed after every occurrence of the LPS and occasionally after the MPS. The frequency of renormalization following the occurrence of the MPS is related to the value of Q_e. Large values of Q_e, i.e., $Q_e = 0.5$, require one MPS renormalization to occur for every LPS renormalization. When $Q_e \ll 0.5$, two MPS renormalizations must be performed for each LPS renormalization. For very small values of Q_e, three MPS renormalizations are performed for each LPS renormalization. [Ref. 9]

2. Progressive Coding

The intent of progressive encoding is to send the image gradually, in discrete steps, adding more detail with each successive scan of the image. This allows the receiver to build the image beginning with low spatial frequencies and working up to high spatial frequencies. The number of discrete steps required is determined by the encoder. Each step doubles the resolution by employing a default algorithm in the encoder that reduces both horizontal and vertical resolution by a factor of two [Ref. 6:p.3]. The base layer uses the sequential template (details are discussed below), and the remaining layers use

a composite template made up of the previous and current layer. The composite template uses the lower resolution layer as an index to determine the difference values required for encoding. The difference values are computed by taking the difference between the current value and the indexed value from the previous layer, and a prediction algorithm is used to predict the current value.

Deterministic prediction and sequential mode prediction are the two prediction algorithms used. The deterministic prediction permits some pixels to be predicted with no error from previously encoded data. This eliminates .the requirement to encode that pixel and replaces it with the location of the identical one previously encoded. The result is a small improvement in compression performance, but the algorithm is difficult to implement [Ref. 8:p.320]. Sequential mode prediction encodes the image line-by-line. If the current line is identical to the previous line, then a special symbol is encoded that instructs the decoder to repeat the last line.

The decoding process may be done sequentially or progressively. Sequential decoding is done top-to-bottom by completely decoding each preassigned strip (determined by the encoder) at all resolution layers before moving to the next strip. Progressive decoding is the inverse of progressive encoding. All the strips are decoded at one resolution layer before decoding the next layer.

3. Sequential Coding

Unlike progressive coding, sequential coding only transmits the data in the base layer, i.e., the resultant image is of the same quality as the first stage progressive mode. This coding scheme uses a sequential template with options for adaptive templates and sequential mode typical prediction. Figure 6 provides an example of a sequential adaptive template [Ref. 8:p.321]. The box marked "A" is the adaptive pixel, capable of being moved within previously encoded pixels "X" on the template to provide a better prediction of the current sample "?" [Ref. 8: p.322].

This coding method usually outperforms the progressive coding because the information containing the higher spatial frequencies is not encoded. The increase in performance is due in part to only the base layer being encoded and transmitted. Another

Figure 6. Example of an Adaptive Sequential Template.

contributing factor is the resolution reduction algorithm being optimized for the base layer, which ensures acceptable image quality using sequential coding. The trade off is that sequential coding produces a lower quality image than progressive coding. The sequential mode template codes the base layer line-by-line from left-to-right moving top-to-bottom using previously encoded pixels to predict the current one. Decoding is done sequentially as described above under progressive coding. The only difference is that there is only one resolution layer for each strip. The JBIG algorithm typically outperforms both Group 3 and Group 4 by approximately 30 % [Ref. 8:p.320].

B. MIL-STD-188-196

This standard was developed to meet established requirements outlined in the National Imagery Transmission Format Standard (NITFS) for compressing bi-level imagery. The bi-level facsimile compression technique for Group 3 is used. It implements run-length code tables for use in Secondary Imagery Dissemination Systems

(SIDS). There are three distinct modes of operation. One dimensional coding (1D) is the first mode. The second mode is two dimensional coding with standard vertical resolution (2DS), and the last mode is two dimensional coding with higher vertical resolution (2DH). All three modes employ the same concept but use a different approach. The concept is to detect run-lengths of one of two colors (black or white) in the image and replace them with their corresponding Huffman codewords. The amount of compression achieved is dependent upon the image being encoded [Ref. 10].

1. One Dimensional Mode

The one dimensional mode is similar to the JBIG sequential coding in that it scans the image line-by-line from left-to-right and top-to-bottom. However, unlike JBIG, the current line is independent of the previously encoded line. This permits each line to be transmitted as it is encoded. Buffer requirements in the encoder are also reduced. To ensure color synchronization, every line begins with a white pixel. If the first pixel is actually black, then an initial white run-length of zero will begin that line, followed by the black run-length. Each run-length of black or white pixels is then encoded using Huffman coding.

a. Huffman Codewords

The standard provides tables of Huffman codewords used in the encoder/decoder. Unlike JBIG or JPEG these default tables must be used; there is no allowance made for implementing custom tables. The tables are arranged by the type of Huffman codewords they contain. Table one contains the terminating codewords that encode run-lengths from zero to sixty-three and "terminate" the codeword. The second table contains the makeup codewords that encode run-lengths from 64 to 1728 in integer multiples of 64. These two tables have separate codewords that determine both run-length and color. The last table contains a set of makeup codewords independent of color. Codewords in this table encode run-lengths from 1792 to 2560 in integer multiples of 64.

28

b. Encoding Procedure

· Each line is encoded one at time, sequentially, beginning at the top of the image. A run-length of less than 64 is assigned its respective terminating codeword from table one. If a longer run-length is detected, then one of the makeup codewords must be used concatenated with the respective terminating codeword. Recall that makeup codewords are in multiples of 64. Therefore a difference will always exist between the actual run-length and that represented by the makeup codeword. The run-lengths that require makeup codewords from table three obtain their color information from the terminating codeword. If this happens to exactly equal the actual run-length, it is concatenated with a terminating zero run-length of the respective color. If "fill" is required, a variable length string of zeros can be appended to the end of the encoded data stream prior to an end-of-line symbol.

2. Two Dimensional Mode

The two dimensional modes are a simple extension of the one dimensional mode with two major differences. Lines in the two dimensional mode are dependent upon the previously encoded line, and this mode also employs a "K" coefficient. This coefficient determines the number of "K-1" lines to be two dimensionally encoded. K may only take on the values, two or four. When K equals two the 2DS mode is implemented, and when K equals four the 2DH mode is implemented.

a. Encoding Scheme

This encoding scheme alternates between one dimensional and two dimensional coding to compress the image. The first line is always encoded using the one dimensional mode as described above. The subsequent K-1 lines will then be encoded using the respective two dimensional mode, followed by a single line coded one dimensionally. This 1D and 2D cycle continues until the entire image has been encoded. The two dimensional mode uses the previously encoded line as a reference line to encode the current line. This is accomplished by encoding only the "changing pixels" with respect to the corresponding pixel in the reference line. A changing pixel is the one directly following a run-length of the opposite color on the same scan line [Ref. 10:p.18].

29

This encoding scheme requires five changing pixels as described below. Figure 7 is a graphical representation of the five changing pixels for the two dimensional coding.
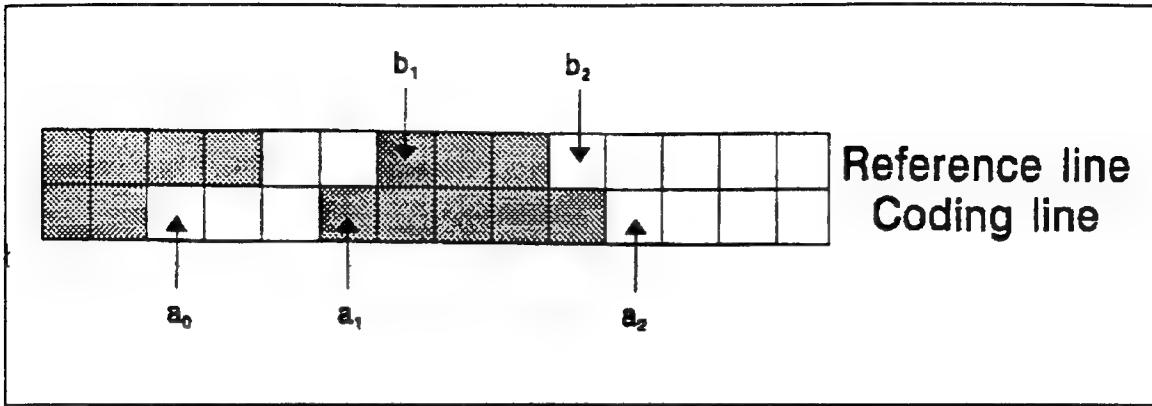
Figure 7. Example of changing pixels from [Ref. 10:p.18].

The five changing pixels can be defined as follows: [Ref. 10:p.18]

1. a_0 is the first reference pixel or starting changing pixel on the coding line.

2. a_1 is the next changing pixel to the right of a_0 on the coding line.

3. a_2 is the next changing pixel to the right of a_1 on the coding line.

4. b_1 is the first changing pixel on the reference line to the right of a_0 and of opposite color, i.e., same color as a_1.

5. b_2 is the next changing pixel to the right of b_1 on the reference line.

Each of the two dimensional coding operations has three different modes that may be implemented depending upon the location of the changing pixels. The first is the Pass Mode that exists when b_2 lies to the left of a_1. This mode has a single unique codeword. Figure 8 provides the relative location of the changing pixels defining a pass mode. The Vertical Mode is defined when the coding operation is not in the pass mode and the distance between a_1 and b_1, $d(a_1,b_1)$, is less than or equal to three pixels. This results in one of seven vertical mode code words, one for each distance: L3, L2, L1, 0, R1, R2, R3, where L represents left and R right. Refer to Figure 9 for a graphical

representation of the relative locations of the changing pixels defining the vertical mode. The final mode is the Horizontal Mode that exists when the coding operation is not in the pass mode and the distance between a_1 and b_1, $d(a_1,b_1)$, exceeds three pixels. This is the 2D mode that employs the Huffman codewords described under the one dimensional coding. Codewords for the horizontal mode are a concatenation of the horizontal mode symbol followed by the respective Huffman codewords for run-lengths a_0a_1 and a_1a_2. An example of the relative locations of the changing pixels defining the horizontal mode can be seen in Figure 10.
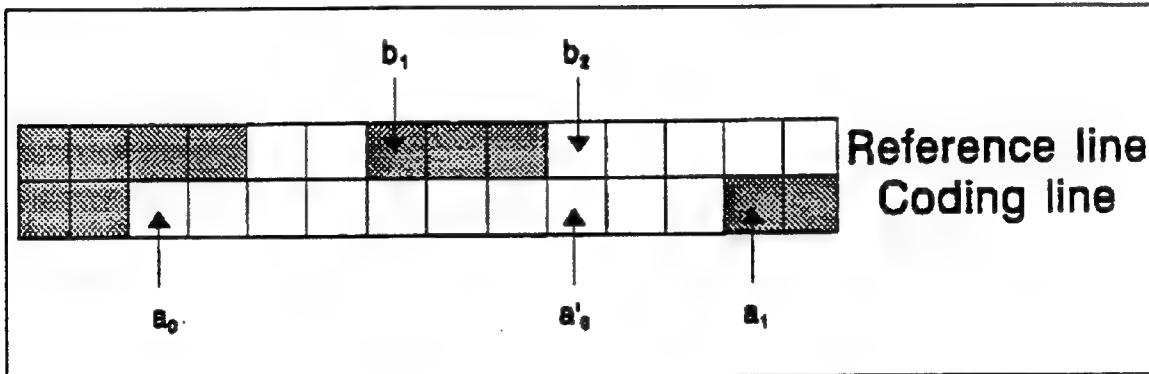
Figure 8. Changing pixels for Pass Mode from [Ref. 10:p.19].
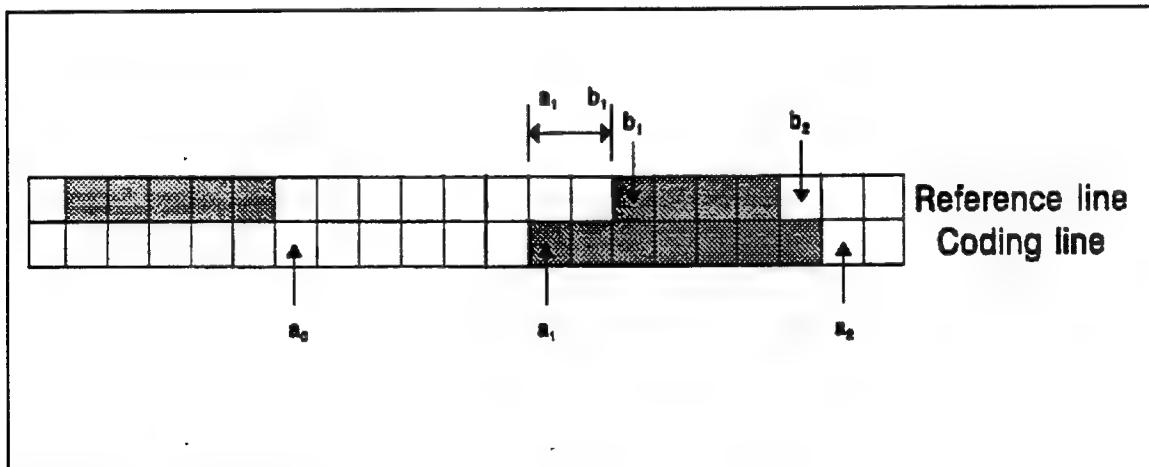
Figure 9. Changing pixels for Vertical Mode from [Ref. 10:p.20].
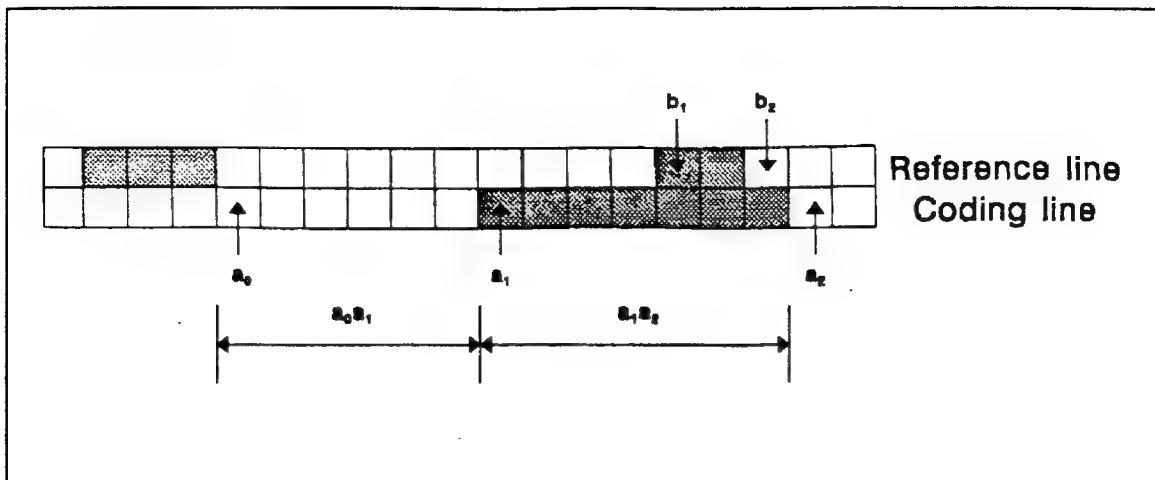
Figure 10. Changing pixels for Horizontal Mode from [Ref. 10:p.21].

b. Encoding Procedure

· The two dimensional encoding procedure is defined in two simple steps. First the algorithm decides if a pass mode exists or not. If it does then the respective symbol is encoded, and the changing pixels are moved as follows. The new location of a_0 is directly below b_2, and the remaining four changing pixels shift to the right relative to the new a_0. Then the procedure for checking for pass mode is repeated. If a pass mode is not detected then step two is executed. Step two is a two part step that first requires the determination of length a_1b_1. If $|d(a_1,b_1)| \leq 3$, then the vertical mode has been identified, and the corresponding code word is encoded. The present location of a_1 then becomes the new location of a_0, and the four remaining changing pixels shift to the right accordingly. If $|d(a_1,b_1)| \geq 3$, then the horizontal mode has been identified, and the corresponding code word is computed using the Huffman tables. The current location of a_2 then becomes the new location for a_0, and the remaining changing pixels shift right accordingly. [Ref. 10:p.21]

In summary, the military standard provides a simple compression scheme that is already in wide use in commercial products. The military standard is based upon the CCITT Group 3 recommendation which employs old technology relative to that employed by JBIG. The JBIG standard employs state-of-the-art technology and typically outperforms both CCITT Group 3 and Group 4 recommendations by 30 percent.

32

Commercial implementations of the JBIG standard are currently available, but it is still relatively new and not in wide use.

IV. GRAY SCALE AND COLOR IMAGE COMPRESSION STANDARDS

Chapter IV focuses on continuous tone still imagery compression standards. Like the previous chapter the first standard described is an international commercial standard—Joint Photographic Experts Group (JPEG). Military Standards 188-197 and 188-198A are discussed in the subsequent sections.

A. JOINT PHOTOGRAPHIC EXPERTS GROUP (JPEG)

The primary objective of JPEG is to compress continuous tone imagery. There are four different modes of operation available in the JPEG standard.   Three of these modes use the Discrete Cosine Transform (DCT) followed by a quantizer resulting in a lossy compression algorithm. The remaining mode is a lossless compression algorithm that can be applied to both continuous tone and bi-level images. The lossless mode uses the Q-coder to estimate the difference values prior to encoding.

The three lossy algorithms are the sequential mode, progressive mode, and hierarchial mode.   All three lossy algorithms exploit the perceptual limitations of the human visual system to achieve increased compression. The sequential mode compresses the image in a single scan and forms the "baseline" algorithm with optional extensions. These optional extensions are provided, so the user can implement one of the other three JPEG modes if desired.   The progressive mode uses multiple scans to compress the image, with each successive scan providing more detail to the resultant decompressed image.   The remaining lossy algorithm is the hierarchial mode that employs downsampling prior to compressing the image. Each time the image is passed through the encoder, it is downsampled by a factor of two both horizontally and vertically. Typical compression ratios of 10:1 to 20:1 are common for all the lossy JPEG algorithms [Ref. 11:p.1]. Most current commercial hardware and software products only implement the JPEG baseline algorithm [Ref. 11:p.1].   A generalization of the JPEG encoder and decoder is provided in Figure 11 .
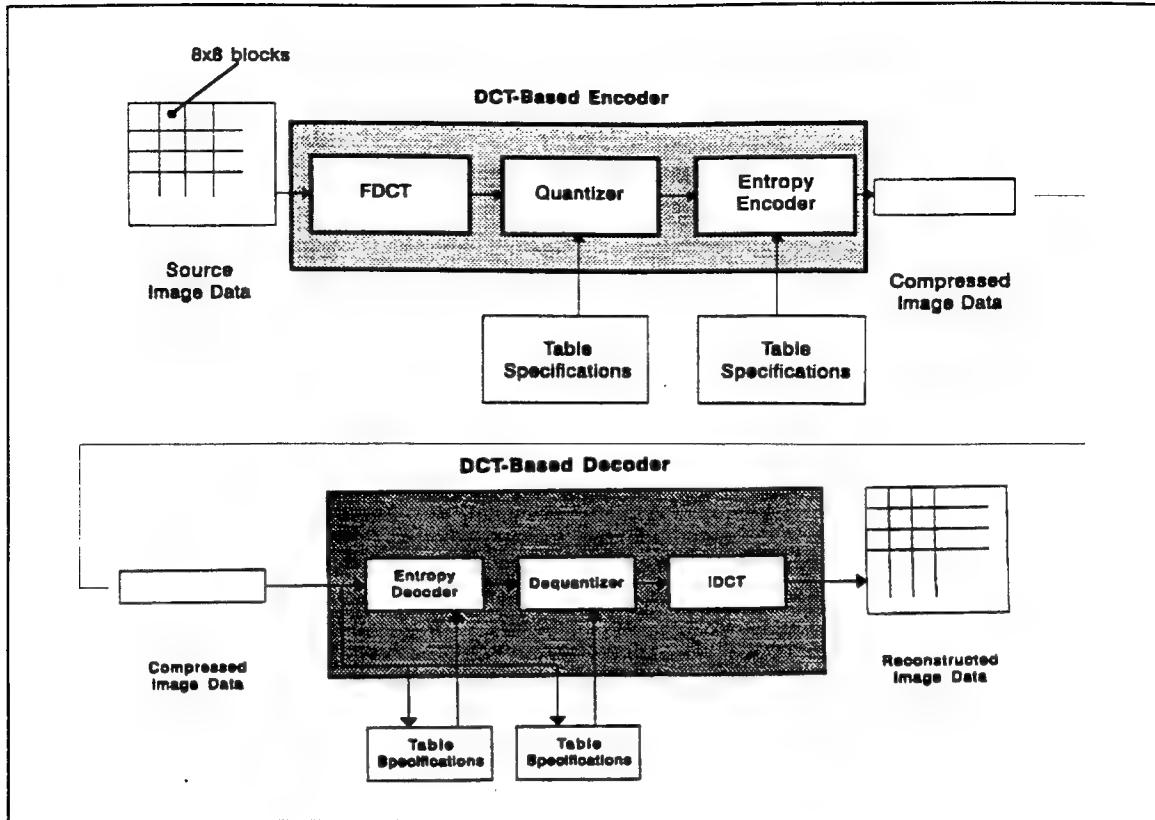
Figure 11. The DCT encoder and decoder after [Ref. 16].

1. The Discrete Cosine Transform (DCT)

The DCT performs a one-to-one mapping between the source image and a frequency domain representation resulting in a 64 point vector [Ref. 12:p.33]. Based upon the discrete Fourier transform, the discrete cosine transform uses a basis set of orthogonal cosine waveforms. Continuous tone image samples can be represented by a weighted linear combination of these waveforms. The given image is divided into 8×8 blocks, and the DCT is applied to each block. A two dimensional DCT is required for these blocks because there are both horizontal and vertical frequency components. This results in 64 DCT coefficients for each block. One DC coefficient represents the zero spatial frequency in the block. Sixty-three AC coefficients are used to represent the various higher spatial frequencies in each block.

Encoding the source image requires use of the forward discrete cosine transform (FDCT). The FDCT transforms the pixel samples in each block into 64 DCT coefficients. The FDCT coefficients are given by [Ref. 8:p.40]

$$S_{vu} = \frac{1}{4} C_v C_u \sum_{y=0}^{7} \sum_{x=0}^{7} s_{yx} \cos\frac{(2x+1)\,u\pi}{16} \cos\frac{(2y+1)\,v\pi}{16} , \qquad (2)$$

where

$$C_v, C_u = \begin{cases} \frac{1}{\sqrt{2}} & , \quad u, v=0 \\ 1 & , \quad else \end{cases} , \qquad (3)$$

s_{yx} represent the pixel samples, and S_{vu} represent the DCT coefficients. The decoder uses the Inverse DCT (IDCT) to transform the dequantized DCT coefficients back into pixel samples, given by [Ref. 8:p.40]

$$s_{yx} = \frac{1}{4} \sum_{v=0}^{7} \sum_{u=0}^{7} C_v C_u S_{vu} \cos\frac{(2x+1)\,u\pi}{16} \cos\frac{(2y+1)\,v\pi}{16} . \qquad (4)$$

A modified version of the FDCT and IDCT is referred to as a fast DCT. The fast DCT uses a faster algorithm that exploits the symmetry characteristics inherent in the DCT by replacing some of the multiplications with additions. This simple change makes the algorithm much faster than the standard DCT algorithm. A detailed description of the fast DCT can be found in reference [8] pp. 53-63.

2. Quantization

Quantization is the only lossy step in the JPEG algorithms. It converts the DCT coefficients into integer values by normalizing by a quantization coefficient (Q_{vu}) and rounding to the nearest integer. Compression is achieved by representing the original image data using the less precise quantized coefficients [Ref. 12:p.33]. The rounding step in the quantization process results in lost information making this a lossy compression algorithm. The JPEG encoder and decoder store the quantization coefficients in tables.

These quantization coefficients are used as threshold levels in the compression process and are based upon the perceptual limits of the human visual system.

The human visual system has two types of perceptual limits. First, differences in brightness are more detectable than differences in color. Second, higher spatial frequencies are less detectable than lower spatial frequencies. Those frequency changes that are hard to detect with the human visual system can have a greater quantization coefficient. Both of these perceptual limits are used in the JPEG standard to determine the proper quantization coefficients that will provide no "perceptual" loss in image quality. Depending upon the desired quality of the resultant image, these quantization coefficients can be altered. Raising them will decrease the image quality, and lowering them will increase quality. The quantization is performed using [Ref. 8:p.A-6],

$$r_{vu} = round\left(\frac{S_{vu}}{Q_{vu}}\right) \qquad (5)$$

and dequantization is performed by the decoder using [Ref. 8:p.A-6]

$$R_{vu} = r_{vu} \times Q_{vu}. \qquad (6)$$

The decoder must have the same quantization tables as the encoder for proper dequantization. If these tables are not known a priori by the decoder, then they must be transmitted along with the compressed image data. Note that this can add significantly to the overhead required for transmission resulting in a lower compression ratio. The decoder will then strip out the dequantization tables for use in image reconstruction.

3. Baseline Algorithm

The baseline sequential algorithm uses the DCT and quantization followed by Huffman coding. Sequential, progressive and hierarchial modes are all extensions of this algorithm and can employ Huffman or arithmetic coding. The six steps making up the baseline algorithm are described below [Ref. 11:pp.1-2]:

38

1. Transform multispectral images into a suitable color space. Coding can be done in the Red-Green-Blue (RGB) color space, or it may be transformed into a luminance (Y) and chrominance (C_bC_r) color space. This step is not performed on gray scale images.

2. Down sample each component by taking the average of groups of pixels. This step is optional and is not performed on the luminance component or gray scale images.

3. Perform a level shift on all the image samples equal to -2^{p-1}, where p is the number of bits. This is done to reduce the precision requirements for the FDCT and to place the DC coefficient at position zero [Ref. 8:p.37].

4. Group the pixel values in each component into the DCT blocks described in section IV:A.1. Sequential application of the FDCT to each block will produce the DCT coefficients required for the next step.

5. Quantize each DCT coefficient using the quantization method described in section IV:A.2.

6. Use Huffman coding to encode the quantized coefficients.

Compressing color images in the YC_bC_r color space will achieve greater compression than using the RGB color space. All three RGB color space components contain both color and brightness information. Only one component (Y) in the YC_bC_r color space contains brightness information. Because the human eye is much more sensitive to changes in brightness, the Y components receive minimal compression. Greater compression can be achieved on components containing only color information (C_bC_r). [Ref. 8]

4. Generalized Encoder and Decoder

The generalized JPEG encoder contains three basic parts: the encoder model, the encoder statistical model, and the entropy encoder [Ref. 8:p.65]. These three basic parts are included in all JPEG encoders with the exception of the lossless encoder. The sequential ordering of the basic parts can be seen in Figure 12 [Ref. 8:p.170]. Recall the baseline algorithm discussed in section IV:A.3 and note that the basic encoder includes

steps 3 through 6 only. Beginning with step 3 the encoder model transforms the source image samples into DCT coefficients. Next the encoder statistical model quantizes the DCT coefficients in preparation for the lossless entropy encoding model. Note that the quantized DC coefficients are differentially encoded between the quantization and entropy encoding steps. The entropy encoding model uses either Huffman coding or arithmetic coding, discussed in the following. Entropy encoding is only performed on the non-zero coefficients with code words extensions describing the run-length of zeros between non-zero coefficients.
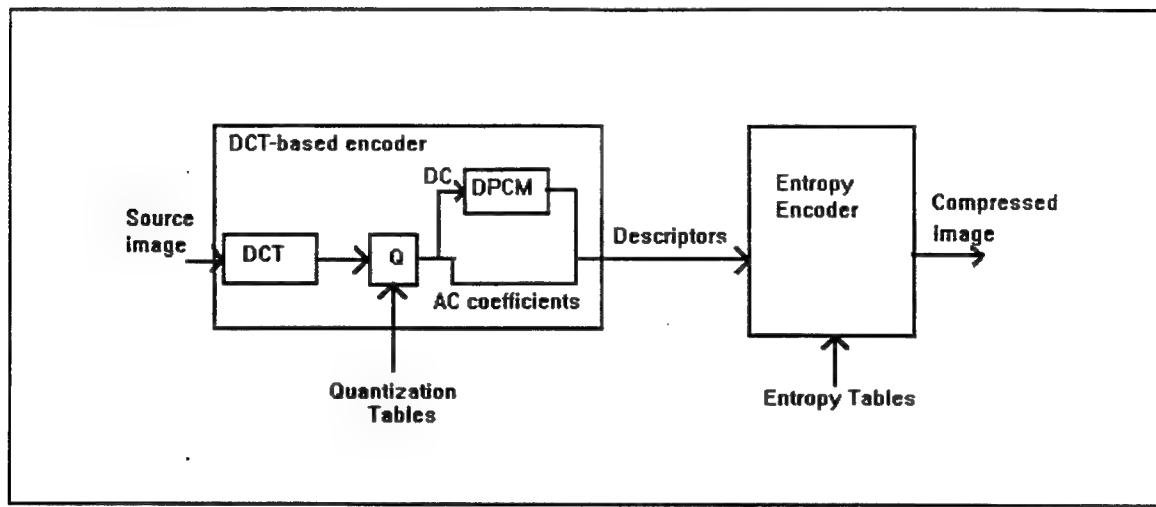
Figure 12. The JPEG encoder model.

The generalized JPEG decoder contains the same three basic parts as the encoder but in reverse order. Refer to Figure 13 for a diagram of the generalized JPEG decoder model [Ref. 8:p.170]. First, the entropy decoder model receives the compressed image data and decodes it. Then the data is passed on to the decoder statistical model for dequantization. Finally, the decoder model transforms the DCT coefficients back into image samples using the IDCT to produce the image. Note that the decoder requires the same tables as the encoder for entropy decoding and dequantization. If these tables' are not known a priori, then they must be transmitted along with the image data increasing overhead and reducing the overall compression ratio.

40
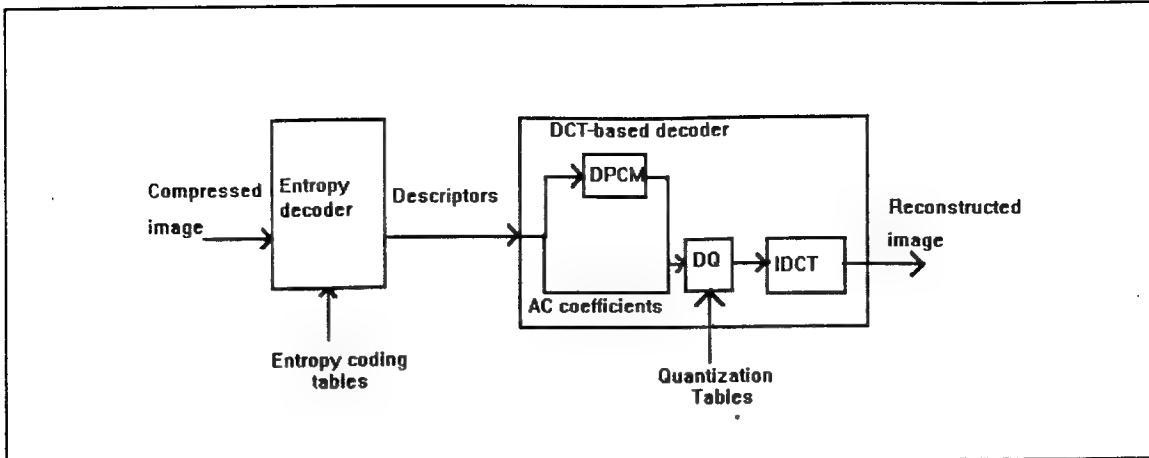
Figure 13. The JPEG decoder model.

5. Sequential Mode

This mode encodes the image in a single scan by sequentially encoding the blocks from left-to-right and top-to-bottom. Each individual block is encoded and then instantly transmitted. Sequential mode uses the baseline algorithm described above with an optional extension to use arithmetic coding in the entropy encoding model. Multispectral (color) imagery may also be encoded using the sequential mode. Each component of the image must be encoded using the sequential mode. The resulting compressed data blocks from each component may be interleaved or transmitted sequentially component by component. Multispectral imagery can be represented in the RGB color space or the YC_bC_r color space. Recall that greater compression can be achieved using the YC_bC_r color space.

Prior to entropy encoding the DC coefficients must be differentially encoded. Differential encoding uses the previously encoded DC coefficient to predict the current one by implementing Differential Pulse Code Modulation (DPCM). This step can be seen in Figure 12. The difference values are then entropy encoded along with all 63 AC coefficients using either Huffman or arithmetic coding. Each coefficient is arranged sequentially, in the order of increasing frequency, in a zig-zag pattern prior to entering the entropy encoding model. First the DC difference value is entropy encoded, followed by the 63 AC coefficients in accordance with the zig-zag sequence illustrated in

41

Figure 14. The lower spatial frequencies being in the upper left corner and the higher spatial frequencies being in the lower right corner. Note that the quantization step has forced many of the higher frequency coefficients to zero.
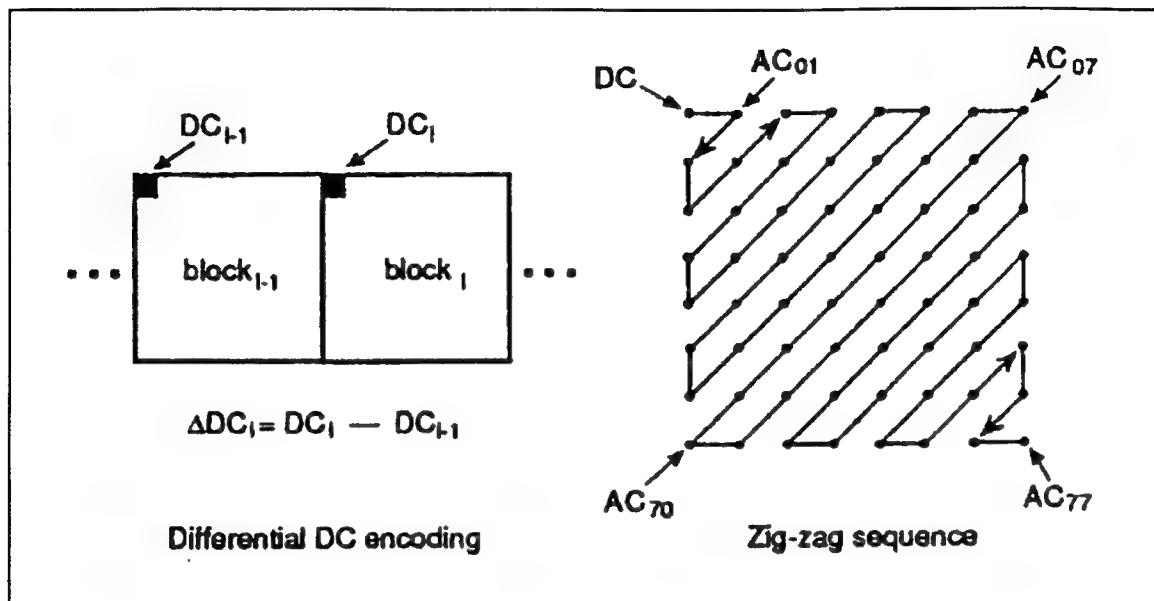
Figure 14. The zig-zag sequence for the DCT coefficients from [Ref. 12:p.35].

6. Progressive Mode

Progressive mode is intended to support real time transmission of imagery [Ref. 11:p.2]. The basic concept is to send the image in discrete increments using multiple scans. It uses the same DCT transforms and quantization as sequential mode. A rough but recognizable image is produced by the first scan, and each successive scan adds more detail to the resulting image [Ref. 12].

There are two ways to implement the progressive mode, and both require an image sized buffer to store the quantized data. Figure 15 provides a graphical representation of both methods. The first method, which uses spectral selection, divides the image into horizontal spectral bands. The initial scan would contain all the data from the top spectral band containing the lowest frequency information. Each successive band contains a higher band of spatial frequencies and provides more detail to the resultant image. The other method uses successive approximation. Successive approximation divides the image vertically sending the first "N" most significant bits (MSB) of each coefficient in the first

42

scan. Each subsequent scan contains the next MSB for each coefficient. The end result is identical regardless of which method is used.

7. Lossless Mode

The lossless mode is the only JPEG mode that does not use the DCT. Lossless mode is an extension of the sequential mode that uses predictive coding. Each pixel is encoded as the difference between the current pixel and the previously encoded neighboring pixels. There are several prediction algorithms that may be used. These algorithms are listed in Table VIII. Selections 1 - 3 are one-dimensional, and selections 4 - 7 are two-dimensional predictors. Selection 0 is reserved for the lossless coding used in the hierarchial mode. The variables in these prediction algorithms represent the neighboring pixels as displayed in Figure 16 [Ref. 8:p.184]. The current pixel being encoded is represented by "x", and the neighboring pixels are represented by "a", "b", and "c".

Each difference value is encoded using the same entropy encoding model described previously. Note also that this mode uses the same DPCM as the previous modes. Lossless mode uses predictive coding for all the coefficients, and the other modes use it only for the DC coefficient. The Q-coder used in the lossless mode is the same one described for JBIG in the previous chapter [Ref. 8:p.320]. This mode achieves a much lower compression ratio than the other three modes, but this is the only mode that guarantees the resultant image will be identical to the original. A typical compression ratio of 2:1 is achieved using the lossless mode [Ref. 12].

8. Hierarchial Mode

The hierarchial mode encodes the image by successively downsampling the source image creating a pyramid of resolution layers. Downsampling the image prior to encoding it can decrease the data to be encoded by as much as 50 percent. This mode represents the image in a hierarchy of multiple resolutions as shown in Figure 17. The top layer containing the lowest resolution and being the first one to be encoded. The higher resolution layers are encoded as differences between the current layer and the

43
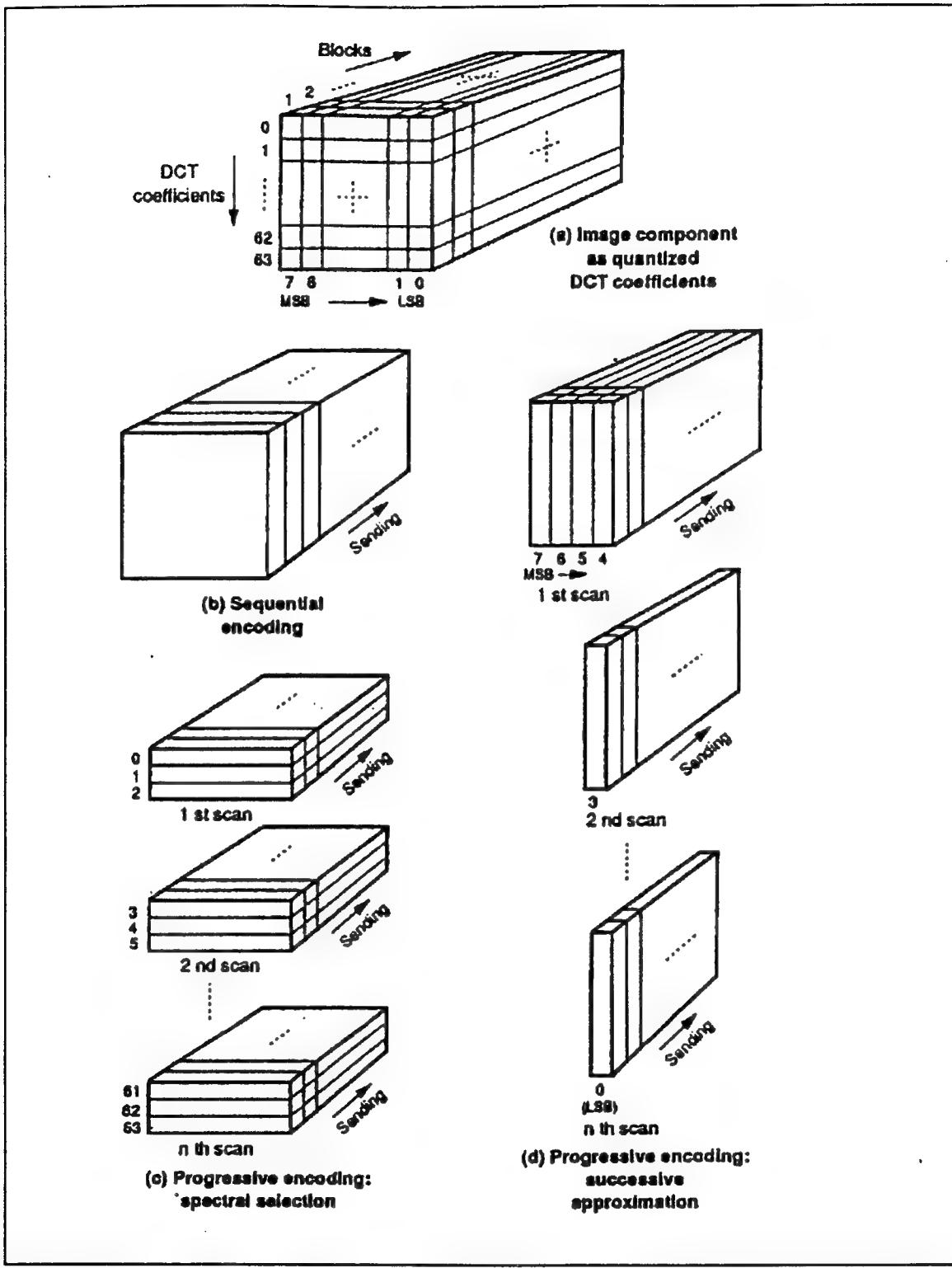
Figure 15. The two methods of implementing the Progressive Mode from [Ref. 12:p.41].

44

Selection Value	Prediction Algorithm
0	no prediction
1	a
2	b
3	c
4	a+b-c
5	a+(b-c)/2
6	b+(a-c)/2
7	(a+b)/2

Table VIII. Prediction algorithms for the Lossless Mode from [Ref. 8:p.184].

previous layer. Wallace [12] provides a concise but clear explanation of the hierarchial mode consisting of the following five steps.

1. Filter and downsample the image by the desired number of multiples of two for each dimension.

2. Encode the reduced image using one of the modes described previously. Note that if lossless coding is used, all subsequent layers will also have to use lossless coding.

3. Decode the reduced image and upsample it.

4. Use the upsampled image as a prediction for the source image and encode the differences using one of the other three modes.

5. Repeat steps three and four until the image has been encoded at full resolution.
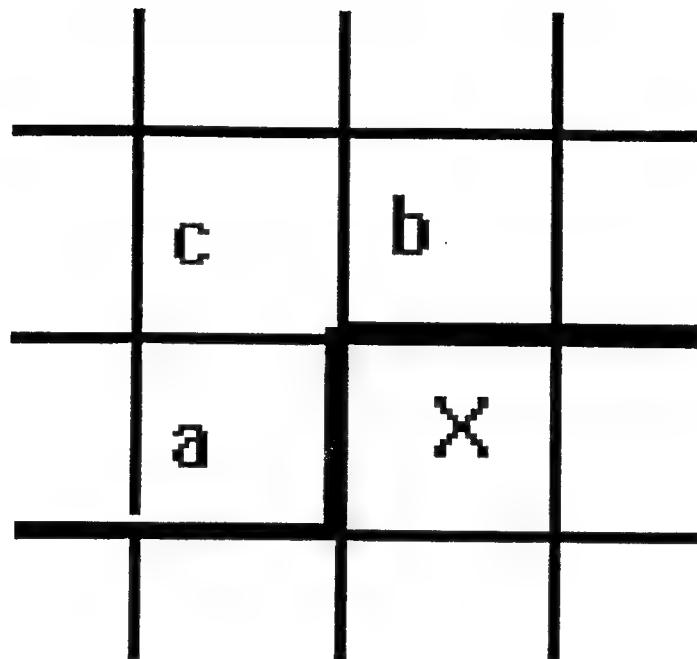
Figure 16. The pixel prediction neighborhood.

Each component in the source image will have a pyramid of resolution layers. The only exception to this is the luminance component because it must be encoded on a gray scale. Non-differential encoding is used on the starting layer for each component. All the subsequent layers in each component use the differential encoding described in step 4 above. This type of differential encoding adds complexity to the encoder and decoder. To perform the differential encoding requires both encoder and decoder to have a temporary buffer capable of storing the entire quantized image. The downsampling and upsampling are performed in either two dimensions (2:1 horizontally and 2:1 vertically) or in one dimension (for example, 2:1 horizontally and 1:1 vertically). For a detailed description of the upsampling filter refer, to the JPEG standard ISO DIS 10918-1, Annex

46
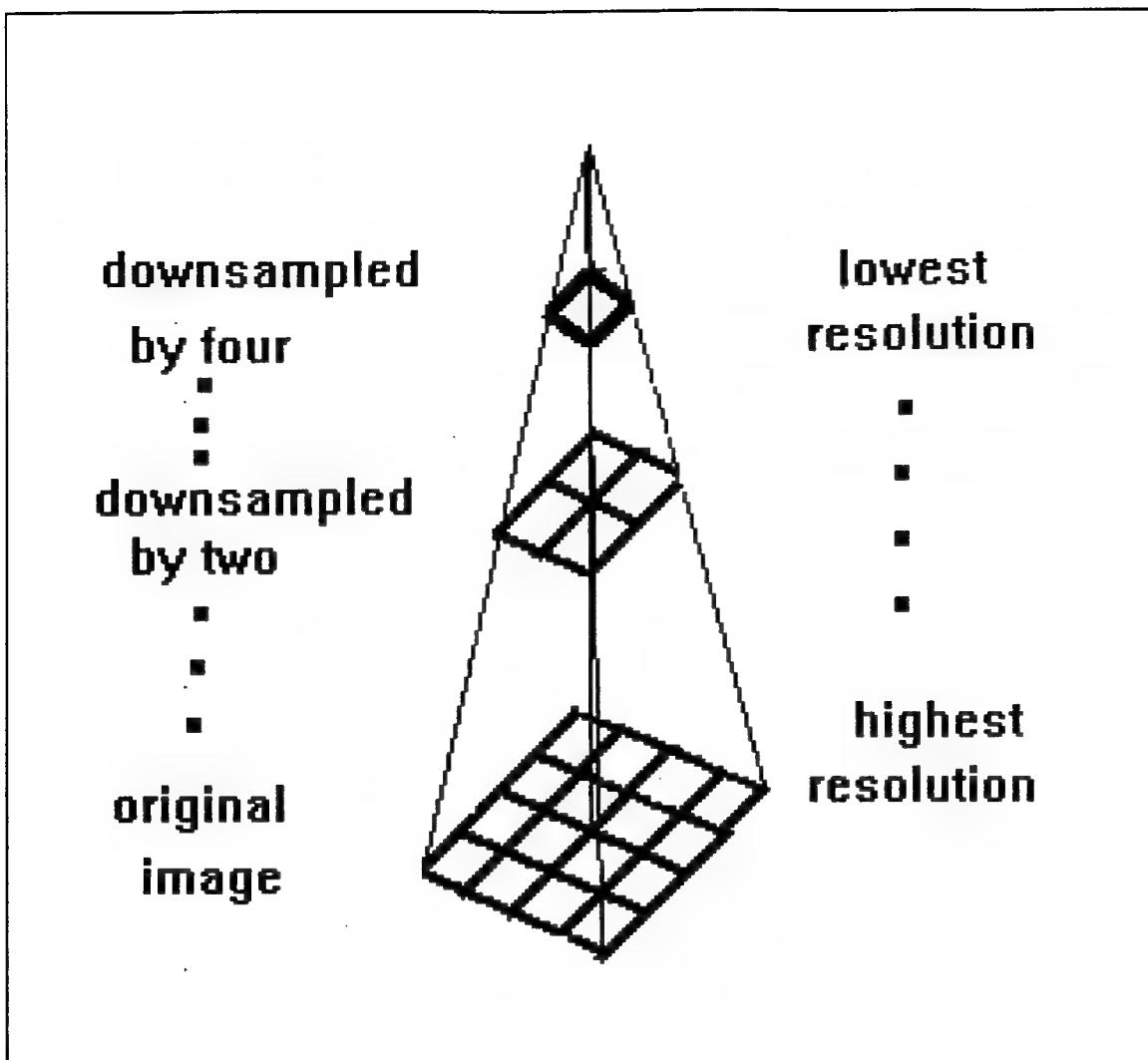
Figure 17. The Hierarchial Mode pyramid of layers from [Ref. 8:p.78].

J [Ref. 8]. This mode requires approximately 33 percent more data to be transmitted resulting in lower overall compression [Ref. 8:p.96].

9. Huffman Coding

Huffman coding compresses data by assigning the least frequent symbol the longest codeword and the more frequent symbols shorter codewords. This simple approach to codeword assignment results in minimizing the average codeword length giving the effect of compression. Huffman coding produces the shortest possible code for binary codewords representing symbols [Ref. 1:p.66]. The length of a Huffman codeword

is a function of its probability of occurrence. The average codeword length can be computed as

$$L_{avg} = \sum_{i=1}^{n} p_i l_i \qquad (7)$$

where l_i is the codeword length, p_i is the respective probability of occurrence of the ith codeword, and n is the total number of codewords. To illustrate the simplicity of this encoding scheme, an example is presented in Figure 18. Referring to Figure 18, note that Huffman coding produces variable length codewords and that no codeword is a prefix of another. The latter part is important because it will allow the decoder to instantaneously decode each codeword as it is received. Huffman coding is described below in 8 simple steps:

1. Arrange the symbols by their respective probabilities of occurrence, placing the most probable at the top and least probable at the bottom.

2. Combine the two least probable symbols by adding their respective probabilities of occurrence together.

3. Take this new probability and create a new probability hierarchy by placing it in its respective location. Note that if this new value is equal to another symbol probability, it must be placed above the old one (this will reduce the variance of resultant Huffman codewords).

4. Continue combining the two least frequent symbols (smallest probabilities) in this fashion until there are only two symbols (probabilities) remaining. At this point the reduction is complete, and the symbol assignment commences.

5. Beginning with the resultant two probabilities, assign one of them the value 0 and the other the value 1.

6. Look at the previous combination step; one of these resultant symbols is a combination of two symbols. One of the symbols making up the resultant symbol will be given the resultant symbol's assigned value with a 0 appended

48

to the end. The other symbol making up the resultant symbol will be given the resultant symbol's assigned value with a 1 appended to the end. The other resultant symbol's assigned value is moved back to the same previous combination step unchanged.

7. Repeat step 6 moving back to next previous combination step until the initial probabilities have an assigned value.

8. The Huffman code is complete, and resulting binary symbols (assigned values) are the Huffman codewords.
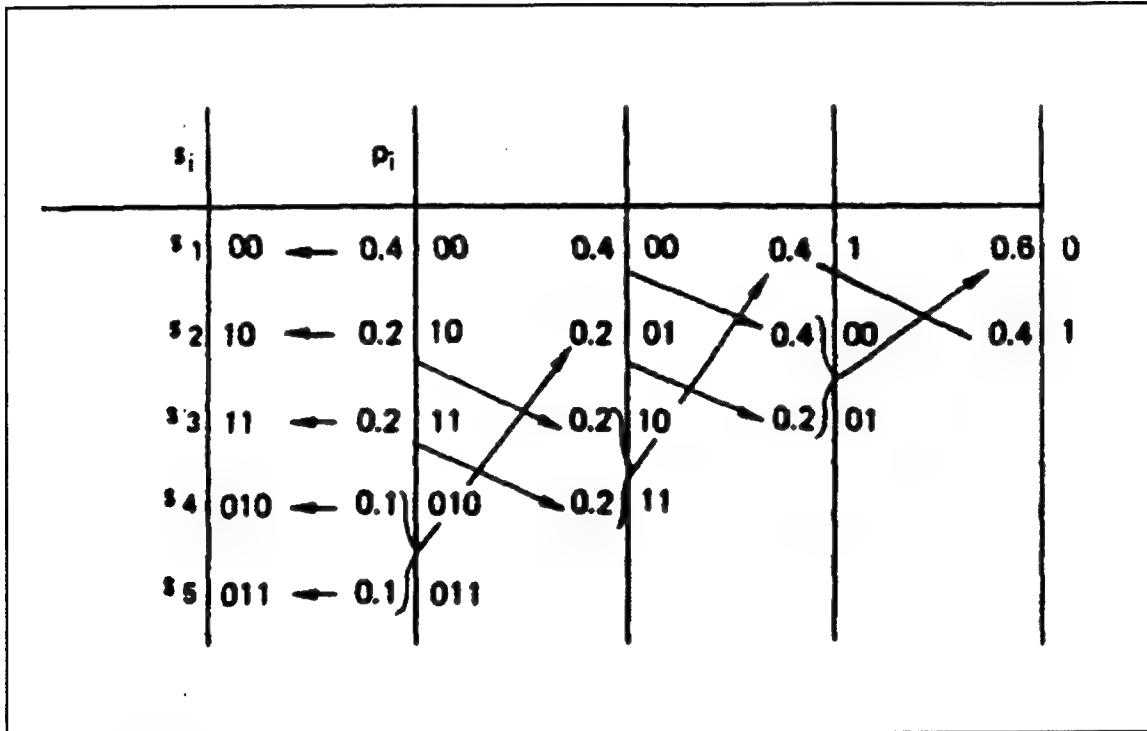
Figure 18. An example of Huffman coding from [Ref. 1:p.67].

Suppose we are given five symbols and their probabilities of occurrence and told to create the Huffman codewords. Figure 18 provides a graphical representation of this example. The left column contains the resulting Huffman codewords and the respective probabilities of occurrence, in descending order. Note how the smallest two probabilities, both equal to 0.1, are combined and moved to the second position (in the next column) creating a new probability hierarchy. This was described in steps 2 and 3 above. This

combination process continues until the two resultant probabilities, 0.6 and 0.4, have been reached. At this point, 0.6 is assigned a binary 0 and 0.4 is assigned a binary 1. Following the arrow back to the previous column, we can see how the 0.6 is split into 0.4 and 0.2, and the new binary values become 00 and 01, respectively. This procedure is repeated two more times until the original probabilities are assigned their new binary values, the Huffman codewords.

The JPEG algorithm implements Huffman coding by precomputing the Huffman codewords and storing them in tables instead of actually implementing the coding scheme online. The same table is required for both the encoder and the decoder. Tables are computed a priori using two different statistical models, one for the DC coefficients and one for the AC coefficients. The sequential mode DC difference values are divided into a set of logarithmically increasing categories. Each category has a unique Huffman codeword assigned to it. A series of bits is then appended to each Huffman codeword providing the sign and exact location within the category. Huffman codewords for the sequential mode AC coefficients combine run-length and magnitude information. Each logarithmically increasing category is combined with each run-length and assigned a unique Huffman codeword. The exact location within the AC category is encoded using the Huffman codewords from the DC difference table. [Ref. 8]

The first stage of progressive mode for encoding DC difference values is identical to the sequential mode. Successive stages send either a zero (no difference) or a one (difference of one with sign indicated by first stage). Encoding the AC coefficients requires an extension of the sequential AC table to include end-of-band (EOB) codewords for encoding EOB run-lengths. Successive stages require two types of encoding. First, the next least significant bit (LSB) is sent from previously known non-zero AC coefficients. Second, new non-zero AC coefficients are encoded using the same process except that they are appended by one amplitude bit and one sign bit. Both the lossless mode and the hierarchial mode use the same Huffman tables as the sequential DC difference encoding. Custom Huffman tables can be used that more closely represent the statistical nature of the image data itself. However, these tables are not known to the

50

decoder a priori and therefore must be transmitted with the compressed data to allow the decoding process to work. This results in a lower overall compression due to the extra overhead required to transmit the custom Huffman table. [Ref. 8]

10. Arithmetic Coding

Arithmetic coding is based upon conditional probabilities and has the ability to adapt to individual image statistics as it encodes. It is also important to note that arithmetic codes are not prefix codes and are not uniquely decodable. Compression is achieved by combining source symbols into a single high precision fraction. The use of a single high precision fraction often results in a precision problem. This precision problem is encountered by most computers and can be overcome in one of several ways. JPEG uses finite precision arithmetic in the Q-coder to overcome the precision problem. The fraction is computed by assigning each symbol a subinterval on the unit interval [0,1). Figure 19 provides an example of arithmetic coding using a tree structure to assign the ones and zeros to each probability of occurrence. [Ref. 3]

Schremp and Weiss [Ref. 13] provide the following simplified description for computing these subintervals used in arithmetic coding:

1. Read in all the symbols and determine the probability of occurrence for each.

2. Arbitrarily assign each symbol a range equal to its probability of occurrence such that all the ranges form the continuous interval [0,1) and are non-overlapping except at the endpoints (e.g., 0.0-0.1, 0.1-0.2, ...).

3. Next compute each symbol's subinterval. The first symbol's subinterval is equal to its range. Beginning with the second symbol, multiply the current message interval by the upper and lower bound of the next symbol and add the results to the lower bound of the previous symbol's subinterval.

4. Continue step 3 until all symbols have been assigned their own subinterval.

5. Each symbol can now be transmitted as a value within its respective subinterval.

The sequential DCT mode using arithmetic coding is again the baseline for extensions into the other modes. First the DC difference values are arithmetically coded followed by the AC coefficients. Pennebaker and Mitchell [8] provide the following description of arithmetic coding. First, the steps for encoding the DC difference values are listed followed by the AC coefficient encoding steps:

1. Is the difference value zero? If yes, then coding is complete; if no, proceed to next step.

2. Encode the sign (positive or negative) of the difference value.

3. Determine if the magnitude of the difference value is greater than, less than, or equal to one. If equal to one, coding is complete, else proceed with next step.

4. Determine the appropriate logarithmic bin (based upon magnitude value) and send respective code word.

5. Append required low order bits to identify exact location of magnitude within the bin.

Note that the AC coefficient encoding algorithm is conditioned on the coefficient index in the zig-zag sequence. This is important because the higher magnitudes are usually in the lower spatial frequencies, and the higher frequencies have magnitudes approaching or equal to zero. The statistical bins reflect this characteristic of JPEG encoding. The following steps are used to encode the AC difference values:

1. Is the AC coefficient index equal to the end-of block (EOB) index? If yes coding is complete, else proceed with next step.

2. Use the same encoding method described for the DC difference values except for the changes noted in the following steps.

3. Is the AC coefficient equal to zero? If yes, increment the index and check the coefficient. Repeat this step until a non-zero coefficient is found.

4. Encode the non-zero coefficient using the methodology described above. When completed, check the index and if it equals the EOB index coding is complete.

5. Otherwise return to step 1 and repeat the entire process.

A simple extension of these two algorithms into two dimensions provides the arithmetic coding algorithm for both lossless and hierarchial modes. Progressive mode requires a slightly more complex extension. The first stage of progressive mode is identical to the method described above. Subsequent stages differ for both the DC difference values and the AC coefficients. All subsequent stages involving the DC difference values simply send the next low-order bit using a binary decision for the odd or even value. Two additional changes are made to subsequent stages for the AC coefficients. First, a check is added to bypass the EOB test for all indices less then the EOB index of the previous layer. Second is a modified coding algorithm for the non-zero coefficients. The modified coding algorithm sends the next low-order bit for previous non-zero coefficients; and new non-zero coefficients, present in this layer but not the previous, undergo the AC coding process described above.

An example of arithmetic coding is provided in Figure 19. We are given eight 3-bit binary symbols and their respective probabilities of occurrence as shown in the first and last columns of the table in Figure 19. Looking at the tree above the table, we have at the very top the symbol I_2 followed by a subinterval, and the symbol $u_0=1$. The symbol I_2 represents the initial 3-bit binary symbol 111. The subinterval following I_2, [37/64, 1) is used to represent the probability of occurrence of I_2 on the interval [0, 1). Finally, the symbol $u_0=1$ is the arithmetic codeword assigned to I_2. This tree represents the arithmetic code for the initial eight symbols and shows where each codeword is represented on the unit interval. The tree also provides the arithmetic codewords as shown in the middle column of the table.

$x_2 = 1$ $I_2 = [37/64, 1) \subset [1/2, 1), u_0 = 1$

$I_1 = [7/16, 1)$, no release

$x_1 = 1$ 0 $I_2 = [7/16, 37/64)$, no release

$I_0 = [1/4, 1)$, no release

0 1 $I_2 = [19/64, 7/16)$, no release

$x_0 = 1$ $I_1 = [1/4, 7/16), u_0 = 0, \; u_1 = 1$

0 $I_2 = [1/4, 19/64), u_2 = 0, u_3 = 0$

1 $I_2 = [7/64, 1/4)$, no release

$I_1 = [1/16, 1/4)$, no release

$x_0 = 0$ 1 0 $I_2 = [1/16, 7/64), u_2 = 0, u_3 = 1$

$I_0 = [0, 1/4), u_0 = u_1 = 0$

1 $I_2 = [1/64, 1/16)$, no release

$x_1 = 0$ $I_1 = [0, 1/16), u_2 = u_3 = 0$

$x_2 = 0$ $I_2 = [0, 1/64), u_4 = u_5 = 0$
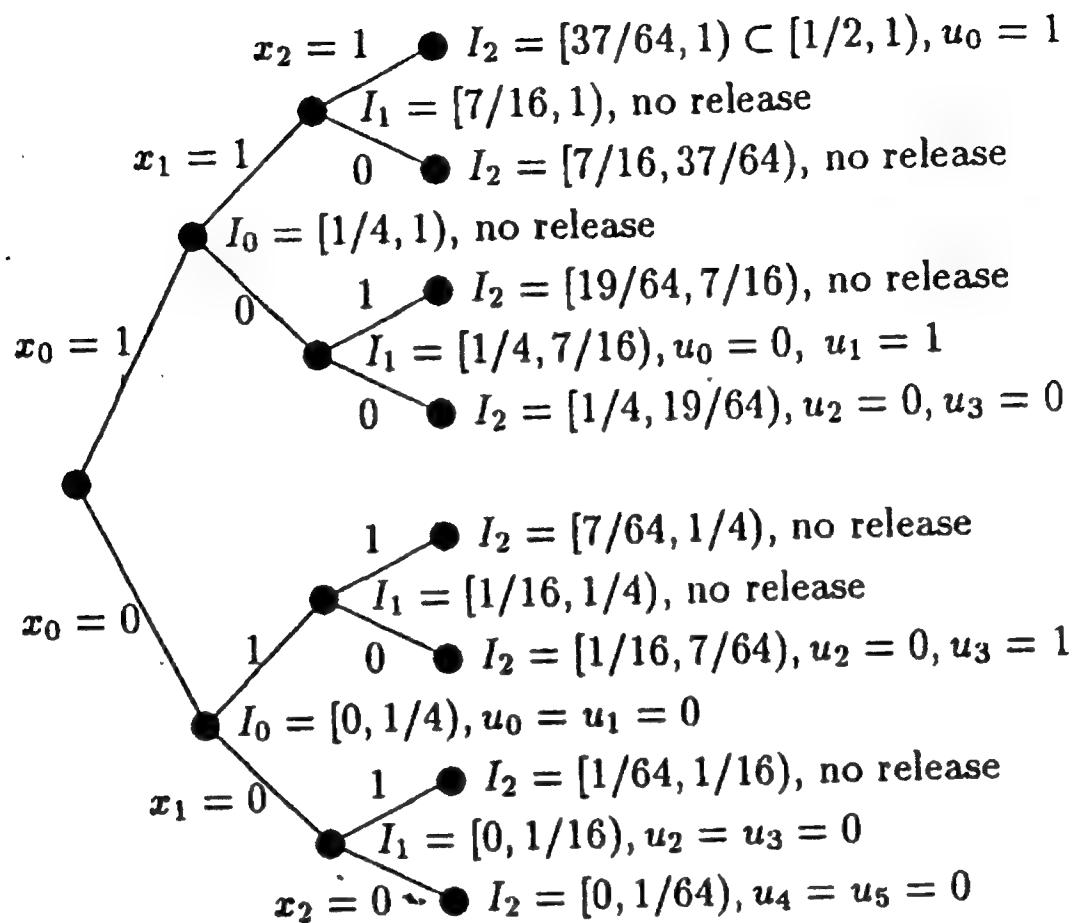
x_0, x_1, x_2	$u_0, u_1, u_2, u_3, u_4, u_5$	l	$p(x_0, x_1, x_2)$
111	1	1	27/64
110		0	9/64
101	01	2	9/64
100	0100	4	3/64
011	00	2	9/64
010	0001	4	3/64
001	0000	4	3/64
000	000000	6	1/64

Figure 19. An example of Arithmetic coding after [Ref. 3:p.281].

B. MIL-STD-188-197

Military Standard 188-197 provides the technical details for encoding both 8-bit and 11-bit gray scale imagery using Adaptive Recursive Interpolated Differential Pulse Code Modulation (ARIDPCM) for use in NITFS systems. In the future this standard may be replaced by MIL-STD-188-198A [Ref. 14], so emphasis is placed on the prospective replacement standard, MIL-STD-188-198A. The ARIDPCM algorithm uses linear and bi-linear interpolation to predict the pixel value and then subtracts the prediction from the actual value creating a difference value. This difference value is then quantized using the default tables provided in the standard and encoded for transmission. [Ref. 15]

There are two types of operations based upon the two sample precisions listed above. Each type of operation has four selectable compression scales. Like the JPEG standard, ARIDPCM divides the image into 8×8 blocks and generates a hierarchial matrix for each block. The difference values at each level of the hierarchy are then quantized and encoded using a fixed length code. An example of the four level hierarchy for a single block is provided in Figure 20. It is important to note that the final hierarchy block is a 9×9 block, with the lower eight pixels of the left column equal to those in the last column, and the top row is identical to the bottom row. Referring now to the 8×8 block defined by the heavy black line, we can see that the block has been encoded into the hierarchy described below.

1. Encoding Scheme

First the image is subdivided into the blocks and then each block is divided into a four level hierarchy forming a prediction matrix. The prediction matrix is made up of one level 1 pixel, three level 2 pixels, twelve level 3 pixels, and 48 level 4 pixels. The prediction matrix is subtracted from the original matrix element by element creating the difference matrix. Quantization of the difference matrix is based upon a "busyness" rating for the block. Busyness is the difference between the maximum and minimum values at level 4. The quantization tables require the hierarchy level, busyness rating and difference value in order to assign a quantized value. Another table, also provided in the MIL-STD,

55

provides the fixed length code using the sample precision, compression rate, and quantized value as inputs. [Ref. 15]
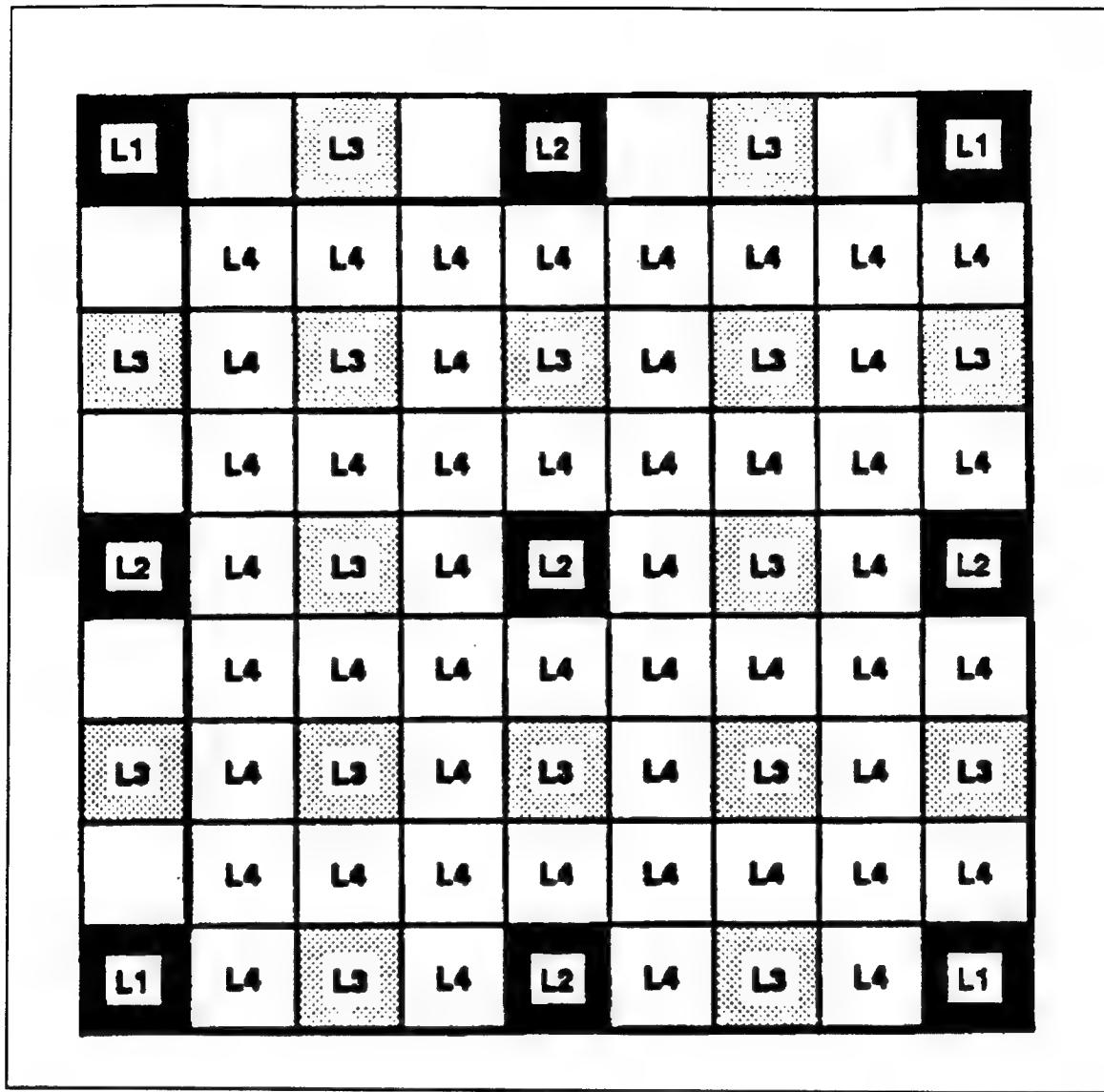
Figure 20. The ARIDPCM block hierarchy from [Ref. 15:p.9].

2. Modes of Operation

Each type of operation may use one of three distinct modes, each of which provide a different result. The mode is chosen based upon the desired resultant image after decoding. To guarantee a specific average bpp over the entire image, the user would choose the driven mode. A specified image quality can be obtained by specifying use

of the non-driven mode. Finally, the composite mode allows regions of interest (ROI) to be encoded using more bpp (lower compression) than the rest of the image. This unique combination of coding allows ROI's to maintain high resolution and still achieve a significant overall compression ratio. Decoding is accomplished the same way for all three modes with the mode information contained in the header. The amount of compression achieved will vary depending on the image and the mode. [Ref. 15]

C. MIL-STD-188-198A

This standard provides the technical details for 8-bit and 12-bit gray scale and 24-bit color imagery being compressed using the JPEG compression algorithm for use in NITFS systems [Ref. 16]. A set of default quantization tables are provided for use in Secondary Imagery Dissemination Systems (SIDS) complying with NITFS. Default Huffman codeword tables are also provided in the standard. The details of this military standard are based upon the JPEG standard described previously in section A. The same DCT algorithm is used, and the encoding and decoding procedures are almost identical. The user may choose to interleave the individual components for multicomponent imagery using the methodology outlined in JPEG. The four modes described in the JPEG standard are also provided for in this standard. However, only the sequential DCT-based mode for 8-bit gray scale imagery in describe in detail.

The remaining three modes (progressive, lossless, hierarchial) are still undergoing revisions and testing. These remaining modes will be added when the acceptance testing is completed. A major difference between this standard and JPEG is the inclusion of the ROI composite compression. Details concerning the implementation of this compression scheme are not provided. Implementation of arithmetic coding and the set of default tables are also incomplete.

In summary, the JPEG standard provides state-of-the-art technology for the compression of continuous tone still images. There are four compression modes provided in the JPEG standard; three are lossy compression algorithms, and one is a lossless compression algorithm. The modes use secondary entropy encoding for further

57

compression, providing a choice between Huffman coding or arithmetic coding. Only the sequential mode is currently available in commercial products and is also the only mode currently described for military applications.

V. VIDEO COMPRESSION STANDARDS

This chapter provides a detailed description of four different video compression standards. The CCITT recommendation H.261 video teleconferencing standard is described first followed by the Moving Pictures Expert Group (MPEG) video compression standard. A brief overview of MIL-STD-188-331 is provided, and the chapter concludes with an overview of Federal Information Processing Standard (FIPS) Publication 178.

A. H.261

The CCITT recommendation H.261 video teleconferencing standard is intended for real time video telephone (VTP) and video telecommunications (VTC) applications using the Integrated Services Digital Network (ISDN). This standard is often referred to as p×64 where p is an integer between one and thirty, and the 64 implies one basic ISDN channel. Video telephone applications require a value of p equal to one or two, and VTC requires $p \geq 6$. The increase in p for VTC is mainly to support the full motion and increased quality requirements. Two different video formats are supported by this standard. Both use the same hierarchial structure but, one requires four times as much bandwidth as the other. [Ref. 17]

The two formats are Common Intermediate Format (CIF) and Quarter-CIF (QCIF). All encoders/decoders (codecs) must operate using QCIF, and CIF operation is optional. Channel capacity often decides the choice of format using QCIF in restricted networks and CIF in unrestricted networks. Another factor that often drives the choice of format is the application. Video teleconferencing applications usually use CIF to ensure higher quality, and VTP uses the QCIF. Both formats have a maximum picture rate of 30 frames/s and may drop one, two, or three frames between transmitted frames [Ref.17:p.60].

There are four levels comprising the hierarchial structure of CIF and QCIF. A single picture is the upper most level of the hierarchy. Each CIF picture is a set of 12 groups-of-blocks (GOB), and each QCIF picture is a set of three GOBs with each GOB

comprised of 3×11 macro blocks (MB). A macro block is made up of four 8×8 luminance blocks and two 8×8 chrominance blocks as shown in Figure 21 [Ref. 17:p.60]. The blocks are similar to the 8×8 blocks used by JPEG for multicomponent imagery.

Figure 21. An example of a macro block .

The video encoder used by H.261 can be broken down into two major categories. First is the source coding that deals with the source data. Lossless entropy coding is the second category. Source coding employs a lossy compression algorithm using one of two modes, intraframe mode and interframe mode. The intraframe mode is used for the initial picture frame and each subsequent frame that contains a "new" scene. Each block in the frame is transformed using the DCT and linearly quantized prior to lossless entropy encoding. Intraframe mode is very similar to JPEG and like JPEG removes the spatial redundancy of the frame. A diagram of the source encoder is provided in Figure 22. Sequences of similar frames are encoded using the second mode, interframe mode.
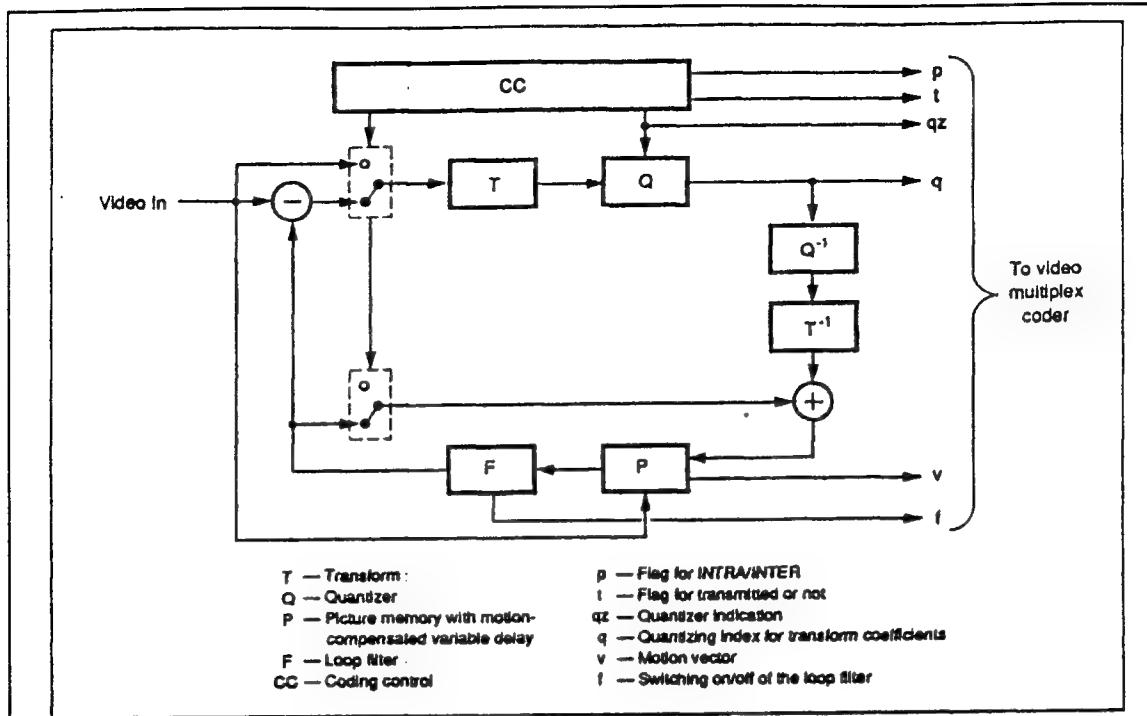
Figure 22. A diagram of the H.261 Source Encoder from [Ref. 17:p.61].

Interframe mode uses DPCM at the MB level on the luminance component comparing the current MB to the previously encoded neighboring MBs from the preceding frame. A difference is computed and compared to a predefined threshold. If the value is below the threshold, nothing is transmitted; otherwise, the difference value is transformed using the DCT and linearly quantized prior to entropy encoding. The quantizer step size is determined by the level of the transmission buffer. As the buffer fills up, the step size increases resulting in less data being transmitted; as the buffer level drops, so does the quantizer step size. This helps maintain a constant data rate while minimizing the buffer requirements. Note that as the quantizer step size increases, the resultant video image suffers an increasing degree of degradation. [Ref. 17]

B. MOVING PICTURES EXPERT GROUP (MPEG)

The objective of MPEG is to provide full motion digitized video capable of integrating text and graphics and to transmit the data over existing computer or

61

telecommunications networks [Ref. 18]. Using 8 bit sample precision, MPEG was designed to optimize the trade off between computational complexity and compression efficiency [Ref. 19]. Below is a list of the four parts that make up the MPEG standard:

1. MPEG System: defines the synchronization of compressed video and audio signals.

2. MPEG Video: describes the video compression requirements, bit stream syntax and decoding process.

3. MPEG Audio: describes the three audio layers used for audio encoding.

4. Compliance Testing: describes the testing procedures for MPEG encoders to ensure that they are MPEG compliant.

Only part two, MPEG Video, is discussed in this chapter. Chapter VI will describe the three audio encoding layers defined in MPEG Audio. The remaining two parts of the MPEG standard will not be discussed because they do not apply to the actual compression algorithms. For details refer to documents IS 11172-1 and IS 11172-4 [Ref. 20].

Currently there exist three phases of MPEG. The first phase is MPEG-1 and is described in detail below. This phase is primarily concerned with video and audio coding at 1.5 Mbps. Video and audio coding at 4-9 Mbps is the focus of MPEG-2. Use of generic coding for video allows MPEG-2 to produce cable television quality video at a fraction of the bandwidth. Both MPEG-1 and MPEG-2 have been implemented using existing VLSI technology. More details describing the second phase are provided later in this chapter. The third phase focuses on very low bit rate video coding for VTC and VTP applications and is referred to as MPEG-4. A fourth phase, MPEG-3, was absorbed by MPEG-2 when it was discovered that MPEG-2 could support the additional coding requirements for high definition television (HDTV) at 20-40 Mbps. [Ref. 20]

1. MPEG-1

This video compression scheme uses two methods of encoding to achieve average compression ratios of 26:1 for video [Ref. 20:p.3]. Intraframe coding is used to remove the spatial redundancy present within each frame by employing the DCT and modified Huffman tables in a manner similar to JPEG. Interframe coding is used to remove the temporal redundancy existing between frames by employing one of two different prediction algorithms: predictive coding or interpolative coding. These two encoding schemes produce three different types of encoded frames, intraframe (I), predicted frame (P) and bi-directional frame (B). To achieve the compression ratio mentioned above, a combination of intraframe and interframe coding must be used.

Intraframe coding is used to support several MPEG features like random access, fast forward and reverse search and editability. Interframe coding is used to support other features like robustness to errors and increased compression. Each method has particular advantages, but to achieve high compression ratios and still obtain quality video a delicate balance must be maintained between them. Intraframe coding is fast and requires little buffer space but only achieves moderate compression. Interframe coding can achieve much greater compression but at a cost of time and storage. The details describing these two coding techniques are provided below.

a. Intraframe Coding

Intraframe coding encodes the frame as if it were a still image completely independent of previous and future frames. Spatial redundancy reduction is accomplished by using the DCT on each 8×8 chrominance block, one C_b block and one C_r block, making up the MB. Once the DCT coefficients are determined, they are quantized using an adaptive quantizer, which can adapt to each individual block by changing the step sizes as required. The high spatial frequencies are encoded using a coarse (large) step size, and the low spatial frequencies are encoded using finer step sizes. Implementing adaptive quantization on a block-by-block basis allows for accurate coding of both sharp and smooth gradients producing improved quality video after decompression [Ref. 18].

63

Each quantized coefficient then undergoes run-length coding using modified Huffman tables to produce a variable length code.

All three types of frames are intraframe encoded with the only difference being the quantizer used. The I frames require accurate encoding of all the low frequency information to ensure a high quality video frame. This accurate encoding of low frequencies is accomplished by using a quantizer with a finer step size around zero. Those frames that use interframe coding, the P frame and B frame, employ a quantizer with a "deadzone" or large step at zero. This is because these two types of frames are mainly concerned with temporal resolution and contain mainly high spatial frequencies that can be coarsely quantized without degrading overall video quality. A view of the two different quantizers can be seen in Figure 23.
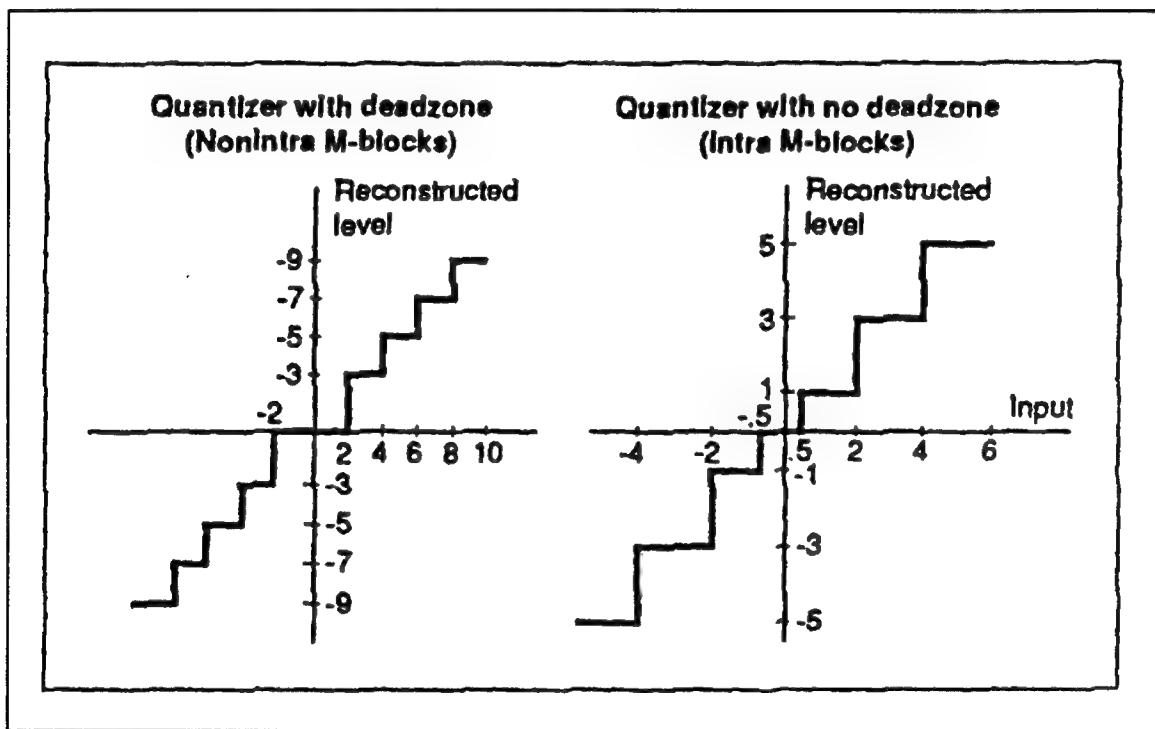
Figure 23. The two intraframe quantizers from [Ref. 18:p.54].

b. Interframe Coding

Interframe coding reduces temporal redundancy between frames using one of two techniques. The first technique is predictive coding. Predictive coding produces

the P frames mentioned above. A prediction algorithm uses previously encoded P and I frames to encode the current P frame [Ref. 18:p.52]. Interpolative coding produces the B frames using a bi-directional encoding scheme that requires both past and future frames. This second interframe coding technique is non-causal and also adds a significant delay (in ms) while increasing the buffer size required for decoding. Another disadvantage of B frames is the increase in computational complexity. However, the huge gain in compression realized using this technique is worth the price. An example of interframe coding can be seen in Figure 24. The sequence of encoded frames is made up of one I frame, followed by three B frames, a P frame, and three more B frames. Recall that the B frames use bi-directional prediction resulting in a non-causal system. [Ref. 18:p.52]
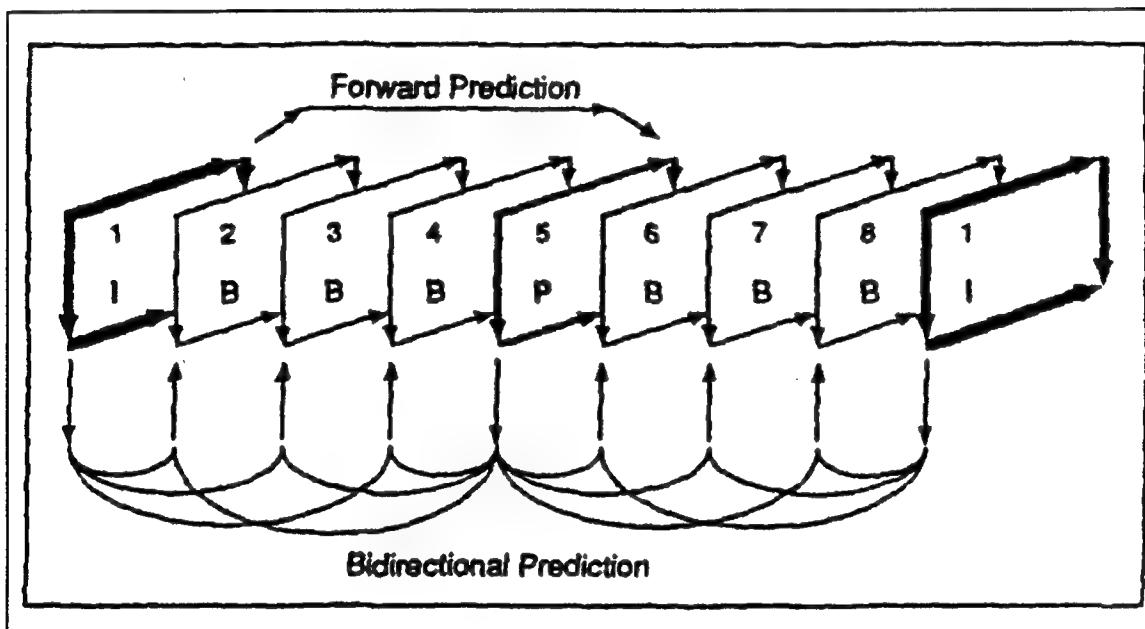
Figure 24. An example of interframe coding from [Ref. 18:p.52].

Both predictive and interpolative coding are designed to represent the motion (temporal resolution) present in each coded frame. The tradeoff for this type of coding is the compression gain against the computational complexity. Predictive coding uses a single motion vector to represent the motion present in each frame. This motion vector may be backward predicted (from future frames) or forward predicted (from previously encoded frames). Figure 24 depicts an example of forward prediction.

Interpolative coding requires both a forward and a backward predicted motion vector. The motion vectors for both techniques are computed the same way.

Motion information in the current MB is differentially encoded with respect to the motion information in the previous MB. This difference motion vector is then intraframe encoded, as described above, further compressing it. Note that MPEG does not specify how to compute the motion vectors; it simply states the number of motion vectors required for each technique. The motion difference vector is computed using motion estimation. Motion estimation attempts to determine the reference MB containing the motion vector that most closely resembles the current motion vector. This will minimize the actual motion difference vector to be encoded.

c. MPEG Bit Stream Syntax

The syntax is very similar to that used by H.261. A layered hierarchy is also used by MPEG with two additional top layers being added to the H.261 hierarchy. The MPEG hierarchy is used to prevent ambiguity and facilitate the decoding process [Ref. 18:p.55]. Below is a list of the six MPEG layers [Ref 18:p.57]:

1. Sequence Layer.

2. Group of Pictures Layer (GOP).

3. Picture Layer.

4. Slice Layer (equivalent to GOB layer in H.261).

5. Macro Block Layer.

6. Block Layer.

The multiple layers lead to several parameters used to define the actual syntax referred to as constrained parameter bit stream (CPB). A set of six parameters, intended to normalize decoder complexity, reduce buffer size and bandwidth and still

address the various applications, makes up CPB. Each parameter and its respective limit are provided in Table IX. These parameters are constrained by the source input format (SIF) for MPEG. The SIF can be defined as a 2:1 horizontal and 2:1 vertical subsampling of the 601 format. The Consultive Committee on International Radiocommunications (CCIR) drafted the CCIR-601 standard defining digital video equipment used for digital television. There are two actual SIF rates used by MPEG as depicted in the fourth row of Table IX. One is 25 frames/s and the other is 30 frames/s, both of which help optimize the MPEG algorithm and support VLSI implementation using current technology [Ref. 21].

pixels/line	≤720
lines/picture	≤576
MB/picture	≤396
MB/sec	≤396×25 or 330×30
picture rate	≤30 frames/sec
bit rate	≤1.86 Mbps
decoder buffer	≤376.8 kbits

Table IX. The MPEG parameter set from [Ref. 18:p.57].

d. MPEG Decoding Process

. The MPEG standard describes a decoding process, not the actual decoder. In order to accurately reconstruct the video signal, this process must be used. Actual implementation is left up to the decoder designer. First the signal is demultiplexed into its various components: motion information, MB type, quantizer step size, encoded coefficients. The quantized coefficients are then dequantized and inverse transformed using the IDCT producing the reconstructed waveform. This waveform is then added to the prediction formed from the reference frames stored in the buffers. This produces the

next frame, and the decoding cycle is complete. Note that the motion header information dictates if none (I frame), one (P frame) or two (B frame) reference frames are required for the decoding process. A block diagram of this simplified decoding process is provided in Figure 25.
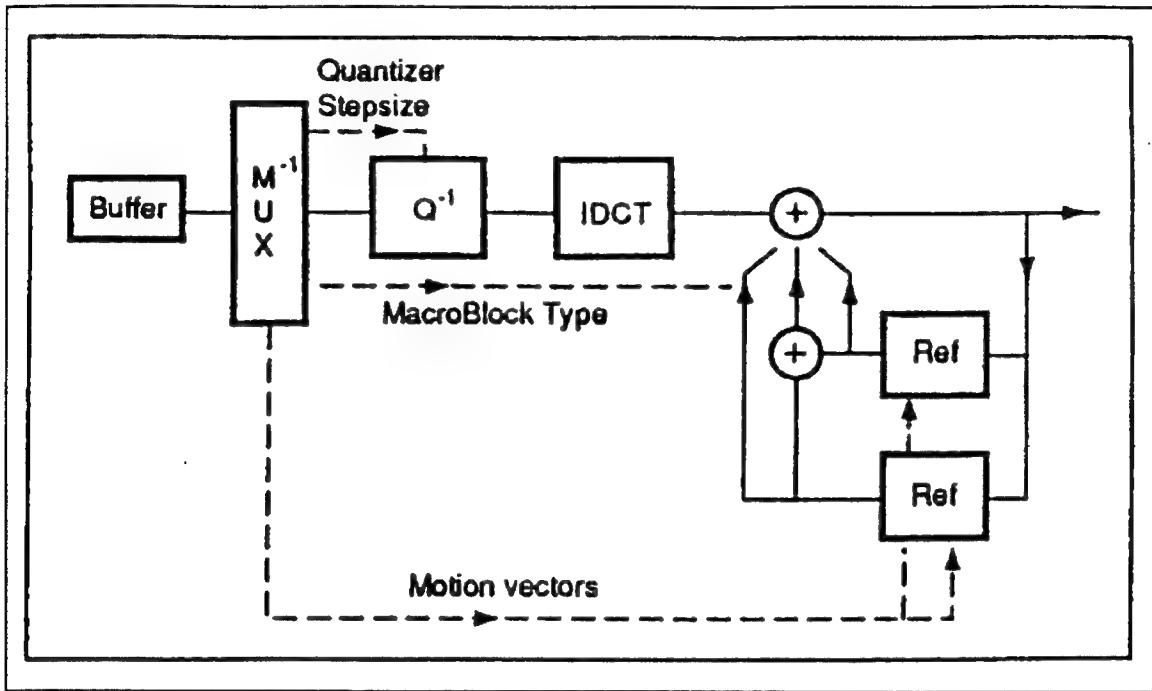
Figure 25. A block diagram of the MPEG decoding process from [Ref. 18:p.57].

2. MPEG-2

Like MPEG-1, MPEG-2 is also a generic method for compressing video data allowing a wide range of applications. The primary application of MPEG-2 is digital cable television (CATV) transmissions using interlaced video data. The inability of MPEG-1 to deal effectively with interlaced video is a primary factor driving the need for MPEG-2. It was also discovered that a few minor changes to MPEG-2 can support the increased data rates and bandwidth requirements for HDTV allowing it to replace MPEG-3. [Ref. 22]

The MPEG-2 standard is divided into two primary subcategories. Main profile and main level (MP@ML) is the first, and high profile is the second. High profile is used for the HDTV applications while MP@ML is used for the CATV applications. Both MPEG-

68

1 and MPEG-2 use the same basic syntax, but the later uses full CCIR-601 parameters resulting in increased video quality. It should be noted that several commercial MPEG-2 encoder chips are already available on the market [Ref. 19]. Below is a list of several improvements made over MPEG-1 that were included in MPEG-2 [Ref. 19]:

1. More aspect ratios were added to the sequence layer providing the increased resolution required for CATV and HDTV applications.

2. The picture layer motion vectors have been refined to half-pixel accuracy.

3. An added parameter allows the user to select one of the following sample precisions for the DC DCT coefficients: 8, 9, 10, 11.

4. The motion vectors in the I frames are concealed making them more robust to errors.

5. The use of non-linear quantization is adopted.

6. The inclusion of four scalable modes: spatial scalability, data partitioning, SNR scalability and temporal scalability, provide added flexibility for use in more applications.

C. MIL-STD-188-331

The purpose of MIL-STD-188-331 is to provide interoperability among different video teleconferencing equipment. Cryptographic and VTC equipment are both covered in detail. Input/output equipment and codecs are not covered by this standard. However, general requirements for video coding are given in addition to referencing H.261 and FIPS PUB 178. Both of the standards referenced for video coding are covered in this chapter. The following chapter will discuss the requirements provided for audio encoding. A list of general requirements for encoding video are provided below [Ref. 23]:

1. The codec must be capable of full motion video for color imagery using at least QCIF resolution.

2. The general requirements described in FIPS PUB 178 must be met.

3. Both forward error correction (FEC) and motion compensation are optional. Note that FIPS PUB 178 requires motion compensation in the decoder.

4. A freeze-frame capability is required. Two options for implementation of this capability are transmitting both full motion and freeze-frame simultaneously or switching between the two modes.

5. Transmission of still imagery is optional. If this capability is provided, it must comply with NITFS using MIL-STD-188-198A. Recall that this requires the JPEG compression algorithm. The only mode required for implementation is sequential DCT based mode with Huffman coding.

D.   FEDERAL INFORMATION PROCESSING STANDARD (FIPS) PUBLICATION 178

Federal Information Processing Standard Publication 178 defines the specifications for video telephony (VTP) and video teleconferencing (VTC) systems [Ref. 24:p.1]. Similar to the MIL-STD discussed above, this standard simply references other commercial standards that must be used. There are two standards listed for video coding. The first is H.261, and the other one is American National Standards Institute (ANSI) T1.314-1991. Both of these standards describe video coding for VTP and VTC applications but with slightly different bit rates. The ANSI standard uses 56 to 1536 kb/s for restricted networks, and H.261 uses p×64 as described in section A for unrestricted systems.

Two types of terminals are described, and a brief list of encoder/decoder (codec) requirements is provided. Class one is a terminal using low data rates for basic ISDN and VTP services with minimum operation defined at p=1,2 using QCIF. The other terminal is class two, and it uses higher data rates providing better quality video and full motion VTC with minimum operation defined at p=6. Class two must also be able to operate at p≥2 for full CIF. Several features in FIPS PUB 178 defining the codec are listed below:

70

1. The codec must be capable of near full motion color at a minimum of QCIF in accordance with H.261.

2. The encoder must be capable of averaging six frames per second.

3. The decoder must average seven and half frames per second.

4. Motion compensation is optional in the encoder but required in the decoder using only one motion vector per macroblock.

This standard does not define any requirements for the audio encoding. Audio encoding is to be accomplished in accordance with ITU-T recommendation H.320. This standard references AV.254 which is the new proposal for CCITT recommendation G.728.

In summary, the only video compression standard in use by commercial industry or the military is CCITT recommendation H.261 video teleconferencing standard. H.261 uses the QCIF four level hierarchy made up of picture layer, slice layer, macro block layer, and block layer. This VTC standard uses either interframe mode or intraframe mode for encoding the source data, followed by lossless entropy coding. The MPEG standard is an extension of H.261 providing increased compression for use with full motion video. The MPEG standard has three phases: MPEG-1, MPEG-2, and MPEG-4. The first two are currently available, and the third phase is still being drafted. Both MPEG-1 and MPEG-2 use the same basic encoding technique, but target different bandwidths. The encoding technique employed by MPEG is a hybrid of the encoding used by both H.261 and JPEG.

VI. AUDIO COMPRESSION

The main focus of Chapter VI is audio compression standards. There are three standards that are described in this chapter. Audio encoding for use with MPEG video encoding will be described first followed by a brief overview of the audio encoding requirements for VTC provided in MIL-STD-188-331. The chapter concludes with a description of MIL-STD-188-113.

A. MPEG AUDIO ENCODING

Moving Pictures Expert Group (MPEG) audio is made up of three audio coding techniques called layer-1, layer-2 and layer-3. All three techniques use high performance perceptual coding schemes based upon psychoacoustical models. Each layer provides increased complexity and performance as well as slightly longer delays resulting in a hierarchy. The higher numbered layers are capable of decoding all lower numbered layers also. Like the video portion of this standard, the audio standard only specifies the decoder and the bit stream format.

Each layer was designed for a specific target bit rate while still providing quality audio. Layer-1 is for high data rates around 192 kbps and layer-2 for medium data rates at 128 kbps. The low data rates around 64 kbps are covered by layer-3. All the layers have the same basic structure and use the same general coding scheme called "perceptual noise shaping" or "perceptual subband/transform coding" [Ref. 25]. The three layers can support 32k, 44.1k or 48k sampling rates providing some flexibility in the bit rates. The lowest bit rate supported is 32k and this applies to all three layers. The highest bit rates for layers 1, 2, and 3 are 448k, 384k and 320k respectively.

The encoder analyzes the spectral components of the audio signal using a filterbank and applies the output of the filterbank to a psychoacoustic model. This model produces the noise level that is barely detectable by the human ear. Then the quantizer allocates bits for coding based upon this threshold noise level to meet both bit rate and masking requirements. The sound quality produced is dependent upon the actual

73

implementation used and should improve with technology [Ref. 25]. Decoding of the compressed data is accomplished by synthesizing an audio signal from the spectral components that were transmitted. A block diagram of a basic MPEG audio encoder and decoder is provided in Figure 26 [Ref. 26].
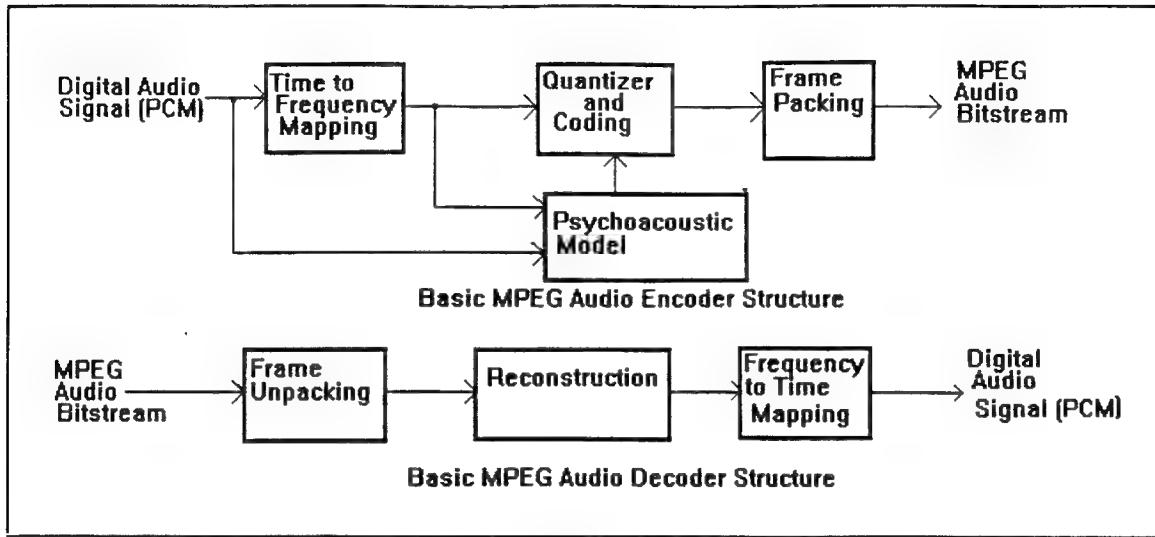
Figure 26. A block diagram of a simplified MPEG audio encoder and decoder.

There are two different psychoacoustic models used, and both serve to calculate the minimum masking threshold for each subband of the encoder filterbank. Masking is when the audio signal blocks or masks unwanted noise from being heard. The noise is produced during quantization by implementing minimum thresholds based upon acoustic properties of the human audio system. Model one is used by both layer-1 and layer-2, and layer-3 uses model two. Both models produce the signal-to-mask ratio (SMR) for each subband which is used to determine the bit allocation. The smaller the SMR the closer the noise ceiling is to the signal resulting in fewer bits required to represent the audio signal. Any information below the noise ceiling is not heard and may be discarded.

Model one uses a fast Fourier transform (FFT) in parallel with a polyphase filterbank. Layer-1 uses a 512 point FFT and layer-2 a 1024 point FFT. The output of the FFT is used to determine the relevant tonals (sinusoids) and non-tonals (noise) in each subband. Tonal information is then used in the model to compute the masking threshold.

74

An individual masking threshold is computed for each frequency line in addition to an absolute threshold. All these thresholds are then added together producing the global threshold for that subband. The minimum masking threshold is then determined, and the SMR is calculated. The SMR determines the bit allocation which in turn effects the degree of compression. [Ref. 26]

Model two uses a 1024 point FFT to calculate the magnitude/phase representation needed to determine the tonal information. Tonal estimation is accomplished using a polynomial predictor that requires the magnitude/phase data from the previous two blocks to predict the magnitude/phase data for each frequency line in the current block. It then normalizes the euclidian distance between predicted and actual values. Each normalized value is logarithmically mapped to a tonal value. The magnitude values are then converted and convolved using a cochlea function. After a small adjustment to offset the DC gain, the masking thresholds can be computed using the tonal information and the convolved spectrum. The SMR is computed in the same manner as above. [Ref. 26]

Layer-1 and layer-2 both use a polyphase filterbank with 32 subbands dividing the audio spectrum into equal spectral bands. Each component is 750 Hz wide providing complete coverage of the human audio range. Layer-1 is a simplified version of layer-2 using a more simplistic encoding scheme. The degree of compression is less than that achieved with layer-2, but the codec delay is shortened, and the resultant quality is improved. Layer-2 provides slightly better performance, sound quality per bit rate, than layer-1 at the expense of added complexity. Layer-2 is currently being planned for use in the digital audio broadcasting network. [Ref. 25]

Layer-3 uses a hybrid scheme that combines the polyphase filterbank from layer-2 with a modified DCT (MDCT). The MDCT is applied to each subband producing increased resolution. It is the most powerful of all the layers providing the greatest audio quality for a given bit rate [Ref. 27]. To further increase compression, the quantized values are Huffman encoded. The Huffman coding employed is similar to that described in Chapter IV but this method uses much smaller tables. Layer-3 was designed for use

with telecommunications networks like ISDN and satellite links. Several important additions included in layer-3 are listed below [Ref. 27]:

1. Higher Frequency resolution providing 576 frequency bands by applying the 18 point MDCT to each filter subband.

2. Increased code efficiency by employing Huffman coding.

3. The addition of a memory buffer to store "extra" data bits until required to reduce or eliminate unwanted artifacts.

4. The inclusion of dynamic bit rate switching allowing more flexibility.

MPEG-2 audio encoding will be backward compatible with MPEG-1 limiting it to the same bit rates used in MPEG-1. However, it adds three more audio channels and also supports up to seven multi-lingual channels. A center channel and two surround sound channels are added to the existing left and right channels. The encoding schemes used by each layer are the same as those used in MPEG-1. Like MPEG-1 audio encoding, this standard also has commercial systems already built and tested. [Ref. 28]

B. MIL-STD-188-331

The audio codec used for military VTC applications may use one of three different modes of operation. Only one of the three modes is required, and the others are optional. Mode zero is a narrow band mode that references ITU-T G.711 and FIPS PUB 178 and is mandatory. The audio subsystem must be capable of operation in mode OF (μ-law). Use of mode OF (A-law) is optional, and use of the two OU modes are not recommended. The next one is mode two defined as wide band audio encoding in an unrestricted network using at least QCIF. Another optional mode is mode three. This is also a wide band audio encoding scheme. The difference between modes two and three is the data rate. Mode two operates at 56 kb/s, and mode three operates at 48kb/s. Both modes two and three are required in the audio subsystem for video encoding at p≥6. A second

narrow band speech mode that operates at 16 kb/s is optional and must conform to CCITT recommendation G.728. [Ref. 23]

C. MIL-STD-188-113

The purpose of MIL-STD-188-113 is to establish interoperability and performance standards for analog-to-digital (A/D) conversion for both long haul and tactical communications [Ref. 29]. A list of five different A/D conversion techniques described in this standard is provided below:

1. 64 kbps Pulse Code Modulation (PCM).

2. 16 kbps and 32 kbps Continuously Variable Slope Delta (CVSD) Modulation.

3. 2.4 kbps Linear Predictive Coding (LPC).

4. 9.6 kbps Adaptive Predictive Coding (APC).

5. 32 kbps Adaptive Differential Pulse Code Modulation (ADPCM).

Each of the modes is described in detail in the following subsections with one exception. Implementation of the 32 kbps ADPCM is not complete and will be added to a future revision of this standard. Due to insufficient details describing the implementation of this technique of A/D conversion, it is not discussed here.

1. Pulse Code Modulation

An input analog signal is bandpass filtered and then sampled at 8kHz converting it to a digital signal. The digitized signal is then quantized into one of eight segments with each segment having 16 levels. A total of 256 levels are used by the compressor/encoder resulting in 128 negative levels, 127 positive levels and a zero level. A view of the positive segments and levels is provided in Figure 27. Note that the 16 levels in positive segment 1 are identical to the 16 levels in negative segment 1, and these two segments are considered to be one segment resulting in a total of 15 segments. Each

77

sample is encoded as an eight bit word and transmitted at 64 kbps in the format described below:

1. Bit zero is the parity bit used to determine if the sample is positive or negative.

2. Bits one thru three define the segment (recall eight segments for each parity).

3. Bits four thru seven define the level within each segment.

The receiver decodes the digital signal using the inverse methodology and low pass filters the analog signal prior to output. [Ref. 29]
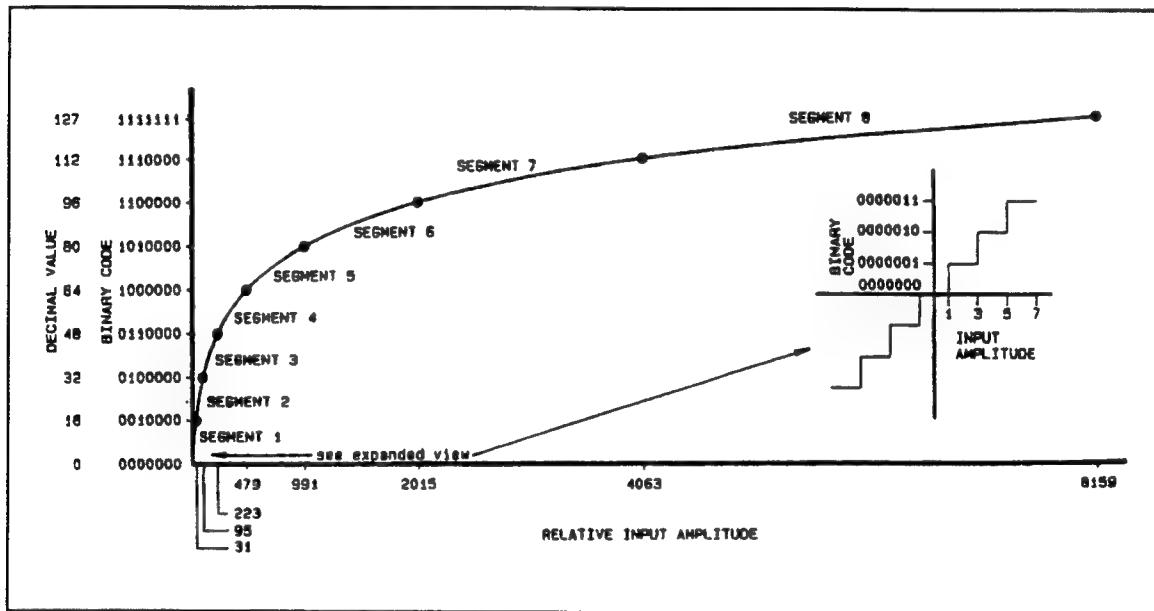
Figure 27. The positive segments for PCM from [Ref. 29:p.47].

2. Linear Predictive Coding

The method of LPC used in MIL-STD-188-113 refers to Federal Standard 1015. All voice encoders using LPC must therefore meet the requirements provided in the federal standard, not the military standard. This coding method is more complex than the previous one due to some additional preprocessing prior to transmission. The input

78

analog signal is bandpass filtered and then sampled at 8 kHz. The digital bit stream is then split into a first order (linear) preemphasis filter, providing the root mean square (rms) amplitude of the signal, and the pitch and voice analysis stage. Here the signal is analyzed to determine if it is noise, voice, or voice-in-transition. If the frame is a voice signal, then the pitch and voice information is encoded, otherwise just voice information is encoded. The rms amplitude is used to compute the reflection coefficients (RC), and both of these are used in conjunction with the pitch information to determine the prediction coefficients (PC). Error control coding is then applied to the PCs for forward error control (FEC), and synchronization bits are added prior to transmission at 2.4 kbps. [Ref. 30]

The LPC receiver uses almost the exact inverse procedure to decode the received digital data. First a serial to parallel conversion takes place, and the synchronization bits are removed. Then the error control bits are checked, and the errors are corrected as required. Parameter decoding and interpolation is performed to obtain the parameters: reflection coefficients, rms amplitude, pitch, voice. The rms amplitude is forwarded to the gain adjustment, the voicing information is sent to the voice switch, and the reflection coefficients are forwarded to the synthesizer. The pitch information is sent to the pitch generator, and then the voice switch chooses either the pitch generator or a noise generator to be input into the synthesizer based on the voice information. The pitch information is synthesized and fed into the gain adjustment regaining proper amplitude. Deemphasis is performed using a first order filter, and then the signal undergoes digital-to-analog (D/A) conversion. The resultant analog signal is low pass filtered prior to being output. [Ref. 30]

3. Continuously Variable Slope Delta Modulation

This technique of A/D conversion uses non-linear modulation via a feedback system requiring a band limited analog signal. Both the 16 kbps and 32 kbps systems use the same encoding/decoding algorithm. Like the previous two techniques discussed, this one also bandpass filters the input to the encoder and lowpass filters the decoder output. The slope voltage is controlled using a run-of-threes binary logic (described in detail

79

below). A difference value is computed by comparing the amplitude of the current sample to that of the previous and encoding the difference (delta).   Unlike most techniques this one outlines the compression ratio that should be achieved. A nominal ratio of 16:1 should be achieved with a maximum not exceeding 21:1 and the minimum greater than or equal to 12:1 [Ref. 29:p.18].

The CVSD encoder first bandpass filters the analog signal and then compares it to the reconstructed analog signal output from the reconstruction integrator.   This reconstructed signal is based upon the last difference sample. These two signals are input to a comparator whose output is a one if the input signal amplitude exceeds the reconstructed signal amplitude and a zero otherwise. The digital data is then sent to the pulse generator, the three-bit shift register and the encoder output. Once the shift register is full, the overload algorithm provides a binary input to the syllabic filter. If three like bits are detected by the overload algorithm, then a one is output to the syllabic filter at the next clock period and a zero otherwise. When the syllabic filter receives a logic one, it produces a charging slope voltage, and a logic zero produces a discharging slope voltage. The pulse amplitude modulator then produces a "continuously variable" signal using the slope voltage to determine the magnitude and the digital input to determine the polarity (one for positive, zero for negative). The reconstruction integrator then integrates the continuously variable signal producing an estimate of the original signal. Refer to Figure 28 for a block diagram of the CVSD encoder and decoder. Note that the decoder is identical to the encoder and is obtained by replacing the comparator with a lowpass filter to produce the final estimated analog signal.

4. Adaptive Predictive Coding

Adaptive predictive coding (APC) is a non-linear prediction method and is the last of the four A/D conversion techniques discussed in MIL-STD-188-113. The analog input signal is bandpass filtered and sampled at 7.6 kHz producing a digital bit stream. A second order preemphasis filter is used followed by pitch processing in parallel with a fourth order two loop iterative predictor used to compute the error estimate. The non-linearity is a result of this fourth order predictor. A final error estimate is then computed

and the prediction coefficients determined. The PCs are then error control coded using a (21,16) Hamming code and transmitted after the addition of the synchronization bits. A block diagram of this encoder can be seen in Figure 29.

Upon receipt of the transmitted digital signal, the APC receiver strips off the synchronization bits and performs error correction as required using the five check bits. The APC receiver uses two prediction filters in series by first predicting the signal amplitude of the original voice signal using the four previously predicted samples and the current PCs. A second prediction filter is then employed to determine the pitch information. The amplitude and pitch are then synthesized and stored in the output buffer awaiting deemphasis. The deemphasized digital signal is then D/A converted and lowpass filtered producing the estimated analog voice signal. A block diagram of the APC receiver is provided in Figure 30.

In summary, the DoD has a standard for A/D and D/A conversion of audio signals, but it does not have a compression standard for audio data. The military standard for A/D conversion of audio signals provides five different methods for performing A/D conversion. The MPEG audio standard is made up of three audio coding techniques that use a high performance perceptual model based upon psychoacoustical models. MPEG layer-3 is targeted for 64 kbps and appears to be the best choice for application to military systems.
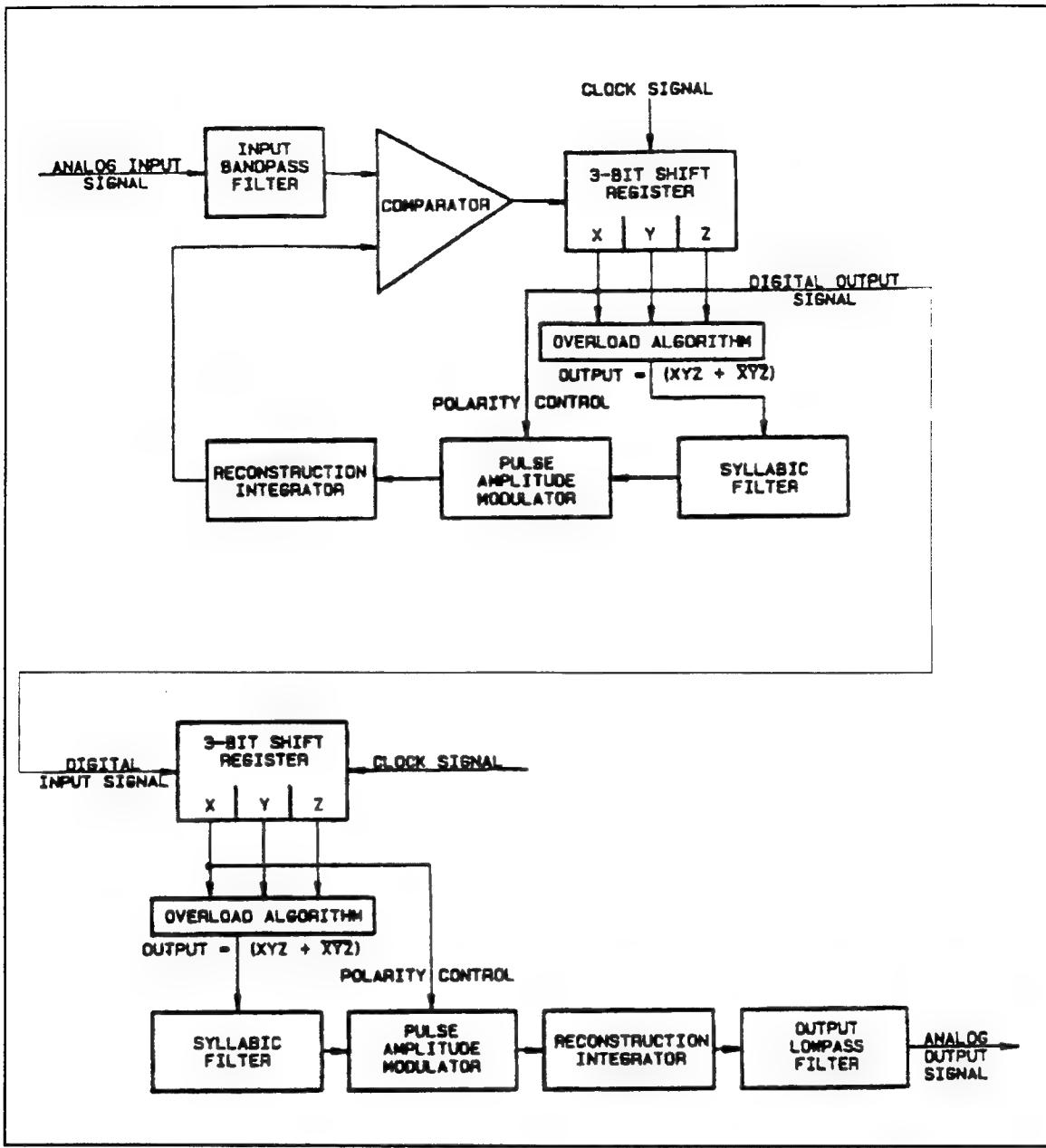
Figure 28. A block diagram of a CVSD encoder and decoder after [Ref. 29:p.19].
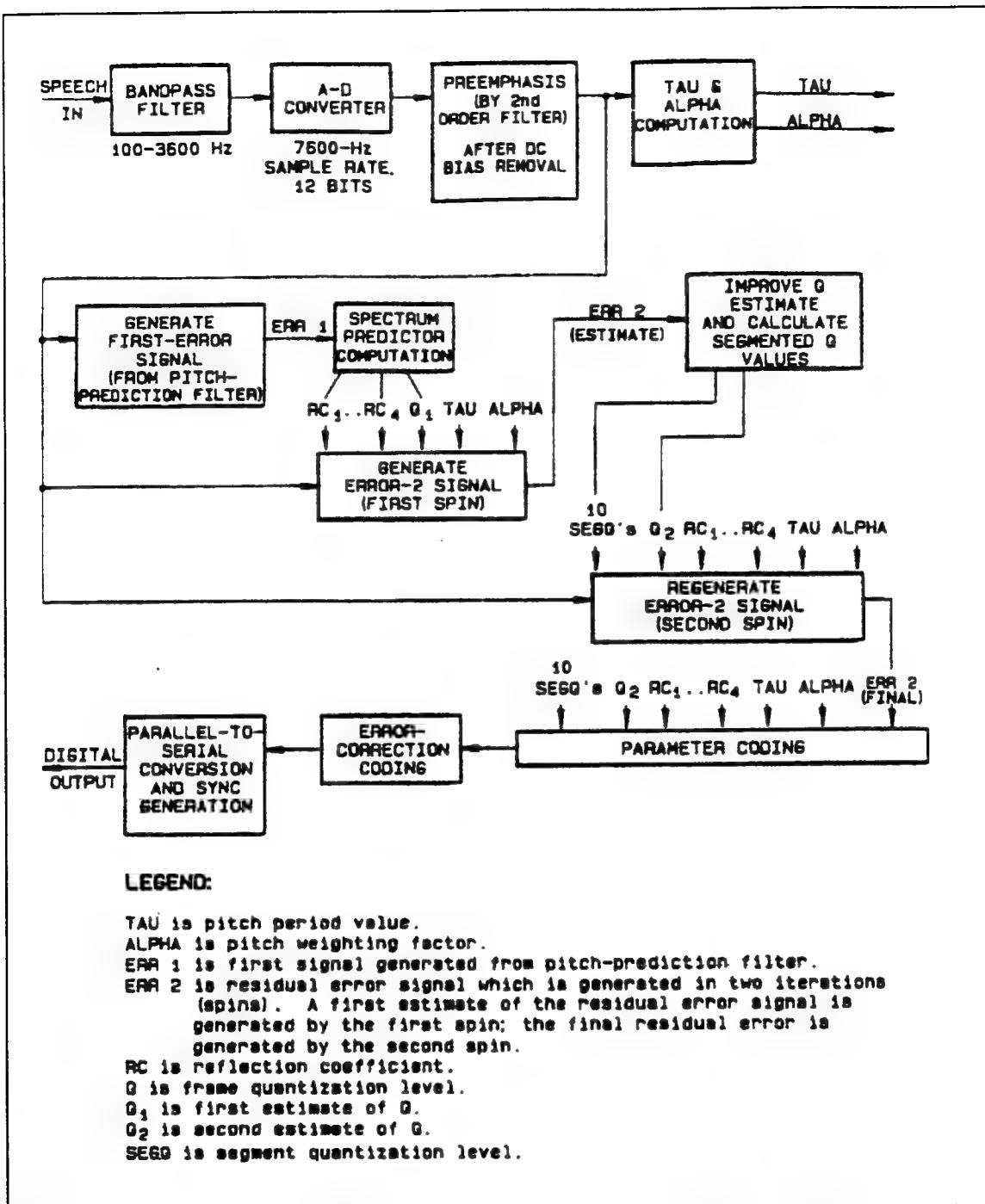
Figure 29. A block diagram of the APC encoder from [Ref. 29:p.28].
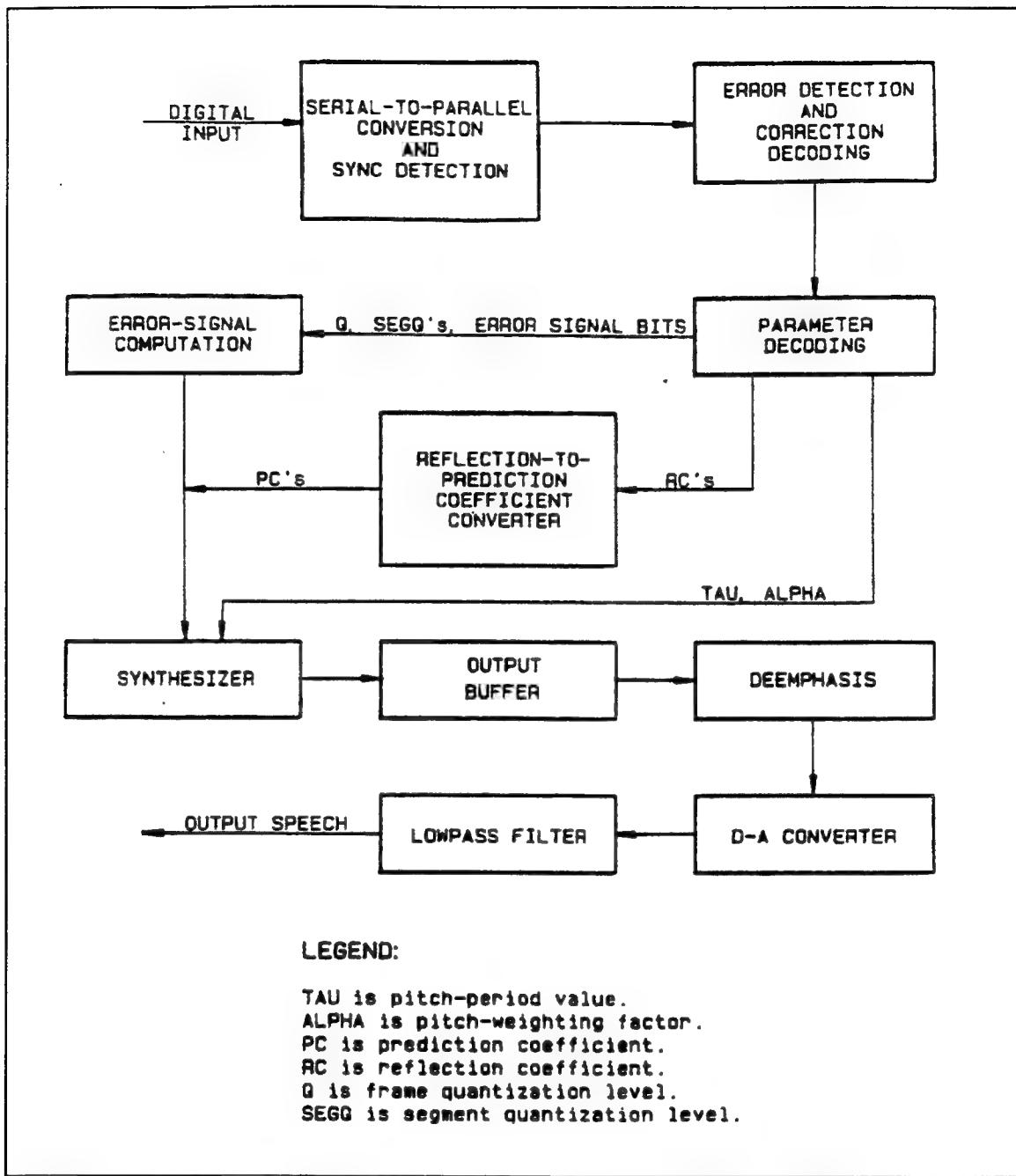
Figure 30. A block diagram of the APC decoder from [Ref. 29:p.29].

84

VII.   EMERGING TECHNOLOGIES AND STANDARDS

A brief discussion of emerging technologies and standards are presented in this chapter.   The two developing compression technologies discussed in this chapter are wavelets and fractals.   A brief overview of asynchronous transfer mode (ATM) and broadband ISDN is also provided followed by an overview of the Multimedia Hypermedia Experts Group (MHEG) standard.

A.   WAVELETS

Wavelets are a set of orthonormal basis functions that produce a set of subspaces. Each subspace contains an approximation of the signal at a different resolution resulting in a hierarchial representation of the original signal.   This hierarchy is made up of subbands contained in one of two signals.   One signal contains all the lower spatial frequency information and the other subband contains the higher spatial frequency information.   The subbands are produced by implementing a set of finite impulse response (FIR) filters.   These FIR filters produce a bank of lowpass and highpass digital filters capable of filtering both digital and analog signals.   The analog signals must undergo A/D conversion prior to the digital filtering process. [Ref. 31]

Wavelets can also be applied to a two dimensional decomposition problem, like a still image, by sampling in both the horizontal and vertical directions.   Like other transform based compression techniques, wavelet transformed signals can be quantized and entropy encoded.   The quantization thresholds are based on the human visual and audio responses and resemble those used in the JPEG algorithm.   Compression and decompression speeds are comparable to that of DCT, but wavelets typically outperform the DCT in the amount of compression achieved for a given image quality [Ref. 32:p.2]. A typical encoder using the wavelet transform technique for image encoding would execute four major steps [Ref. 33].   First, the encoder would perform subband decomposition using the wavelet transform, followed by quantization of the transformed data.   The data is then re-organized prior to entropy encoding.   Compressed data is much

more sensitive to noise because there are fewer bits representing the information. Should an error occur within a compressed file, the resultant error after decompression is usually greater than the same error in an uncompressed file. The wavelet transform provides increased robustness to errors when compared with the DCT [Ref. 31].

B. FRACTALS

Fractals are a promising new technology developed in the mid eighties that employs advanced interpolation techniques. It provides lossy compression through use of iterated function systems (IFS) that are based upon fractal geometry and iterated function theory. An IFS is a set of contractive transformations that map from a defined rectangle in the real plane (no complex plane) into a smaller rectangle. Compression using fractals is slow because of a pairing problem in the algorithm. The decompression is much faster and can provide some degree of image enhancement by repeated iterations. The use of fractals for image compression came about when it was discovered that natural images could be generated from fractals. [Ref. 34]

It was hoped that by reverse engineering the image generation problem, a natural image could be represented by fractals. Natural images contain redundant detail that can be represented with a small number of transformations. This redundant detail is the key to the application of fractal compression for image representation. Jaquin [Ref. 34] was the first to successfully automate a fractal compression algorithm achieving compression ratios up to 50:1 . He used a variant of IFS called partitioned iterated function system (PIFS). The PIFS maps a large rectangle (domain block) into a smaller rectangle (range block) allowing each pixel of the original image to be represented in at least one range block. A pattern of range blocks is produced and is represented by a grid partition. The grid partition contains the range block and associated transformations.

The fractal compression algorithm has problems in the matching of range blocks to domain blocks where the difference between the two must be minimized. More efficient methods of searching the domain blocks are required to increase the compression speed. Entropy encoding can be performed on the resultant grid partition to provide

86

increased compression. Fractals achieve a much higher compression ratio on color images than gray scale due to the chrominance information present in color imagery. When compared with the JPEG algorithm, fractals outperformed JPEG at compression ratios in excess of 40:1, but JPEG worked better than fractals below 40:1 [Ref. 34].

C. ASYNCHRONOUS TRANSFER MODE (ATM)

Asynchronous transfer mode is a two level transport hierarchy providing packet switching for high speed networks. One level is referred to as a virtual channel connection (VCC) and the other level is a virtual path connection (VPC). ATM uses a fixed "cell" size consisting of a 5 byte header and 48 bytes of information. The header contains a header error correction (HEC) byte that is determined by the remaining 4 bytes in the header. Inclusion of the HEC provides header error detection capability and a limited degree of error correction capability. High speed networks require variable data rates and flexible switching control. Asynchronous transfer mode can provide the flexibility and network infrastructure required for high speed networks. [Ref. 35]

A logical ATM connection established between any two network entities is a virtual channel connection. The VCC is the basis of a new high speed network called broadband integrated services digital network (B-ISDN) [Ref. 35]. Each end user may determine their own service quality by using a set of specified "traffic" parameters. These parameters include items like cell loss ratio and cell delay. Two types of switched connections are supported by ATM providing network flexibility. The first type uses call-control signaling, and the other type is a semi-permanent dedicated connection. Type one allows ATM cells to be routed through the network using the most efficient path. Type two is similar to a circuit switched network maintaining a constant end-to-end connection until termination.

A virtual path connection is simply a collection of VCCs with common end points. An example of the ATM connection relationship is provided in Figure 31 [Ref. 35:p.821]. One problem with high speed networks is network control. Creation of the VPC is one method to minimize network control by treating the VPC as a single connection vice

87

multiple VCC connections. This will simplify network architecture and increase network performance and reliability by minimizing the number of connections being monitored by the network management system. Each VPC exhibits the same characteristics as a VCC, plus it reserves one or more VCCs for network management functions. The redundancy between a VPC and VCC is required to support the creation of additional VCCs within a VPC when required with minimal effort. [Ref. 35]
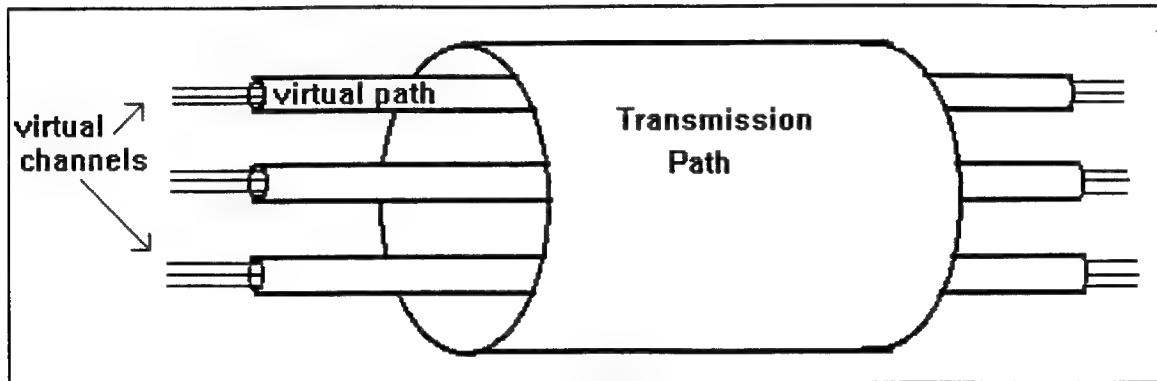
Figure 31. The ATM connection relationship.

D. BROADBAND ISDN

Broadband ISDN (B-ISDN) encompasses all the services and functionality found in ISDN and provides additional services and systems with bandwidth requirements exceeding those of ISDN. An example of typical new services is interactive video and distributed video services. This high speed network provides three types of transmission services. First is the full duplex 155.52 Mbps service followed by the 622.08 Mbps full duplex service. The third service is an asymmetrical channel providing 155.52 Mbps from the user to the network and 622.08 Mbps from the network to the user. Fiber optic cable is the only medium that currently supports this much bandwidth and is one of the driving factors in the implementation of B-ISDN. The user-network interface will be bridged by the introduction of ATM. B-ISDN provides an additional transmission path over the existing narrowband ISDN as shown in Figure 32 . [Ref. 36]

88
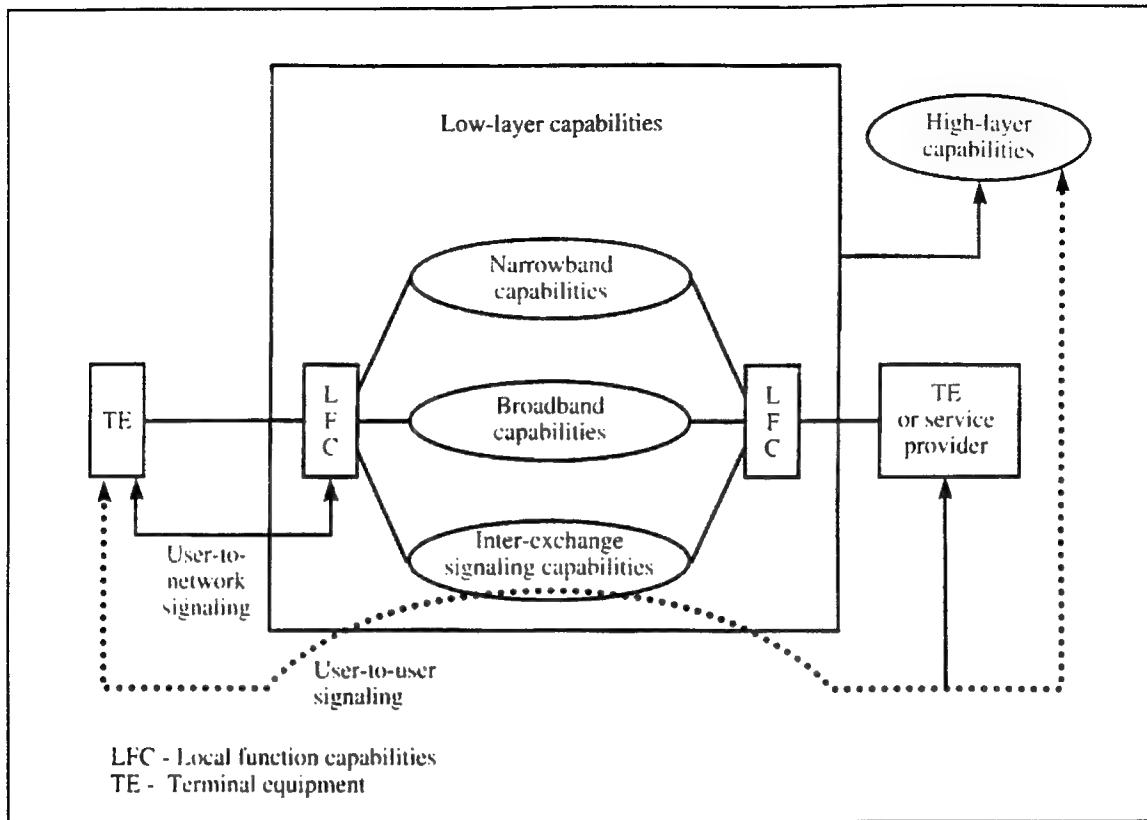
Figure 32. A block diagram of the B-ISDN architecture from [Ref. 36:p.793].

E. MULTIMEDIA HYPERMEDIA EXPERTS GROUP (MHEG)

This standard is based upon an object oriented concept that will cover a wide range of applications and services. It will not replace existing encoding standards like JPEG or MPEG, but it will enhance them by providing a common "envelope" for transmission of data encoded using other standards. The intent of MHEG is to allow different applications to share the same informational resources by providing a common data envelope that will support system interoperability and information exchange. Open system design and portability are key to the successful implementation of MHEG. This is not an encoding standard but is more like a protocol standard. Figure 33 is a block diagram providing the relationship between MHEG and other standards, services and applications [Ref. 37:p.546]. The MHEG standard will provide real time information exchange and presentation that is transparent to the end user. This is similar to the NITF

standard being used by the military in secondary imagery dissemination systems. The difference is NITF only applies to imagery and imagery related data where MHEG will encompass all types of media including video. [Ref. 37]
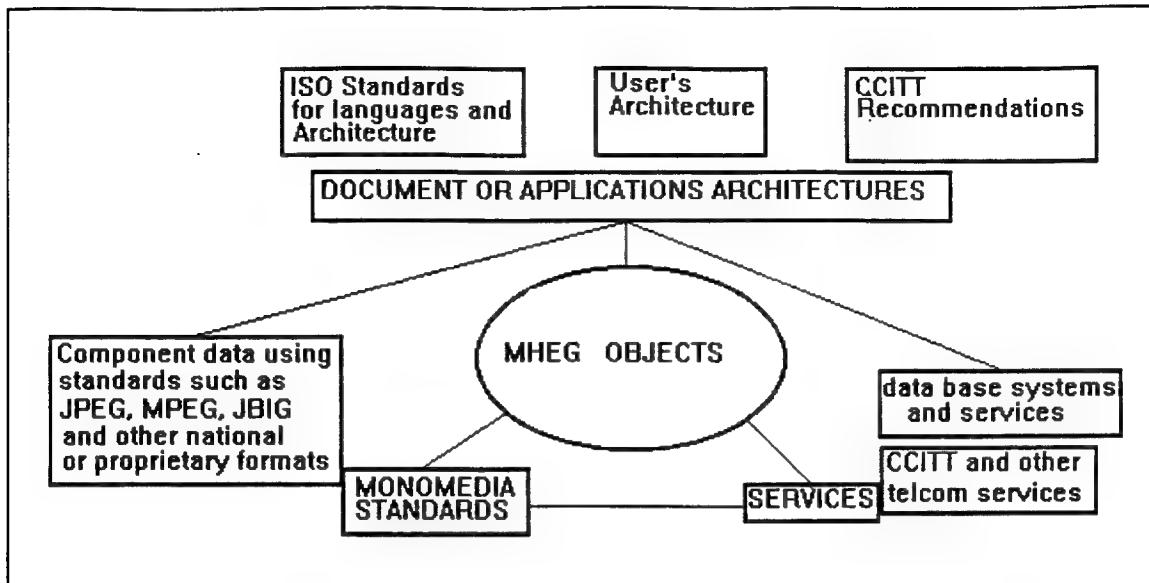
Figure 33. A block diagram showing the relationship between MHEG and other services and standards.

In summary, there are two emerging technologies that could provide increased compression performance, wavelets and fractals. Fractals is still a relatively new technology, and it may take several more years to perfect an efficient compression algorithm using fractals. The introduction of ATM and B-ISDN will provide a substantial increase in high speed network bandwidth supporting the introduction of many new media services. When MHEG is completed, it will provide a common envelope for the transmission of various data types.

90

VIII. GLOBAL COMMAND AND CONTROL SYSTEM

Chapter VIII provides a description of the functionality and objectives of the Global Command and Control System (GCCS). The chapter concludes with a discussion of the applicability of the various standards described in the previous chapters to GCCS.

A. GLOBAL COMMAND AND CONTROL SYSTEM (GCCS)

Current military operations require a fully integrated team from all four service components. This concept of joint warfighting is now at the forefront of the Department of Defense. The need for jointness entered into the DoD limelight following the Grenada invasion and has continued to the present. Prior systems developed for command and control (C^2), communications, or command control and communications (C^3) were stovepipe systems. They were all designed to support the military hierarchy with vertical information exchange. Today's fast paced combat environment requires commanders to communicate both vertically and horizontally. This combined with the commanders' need for imagery and video on the battlefield have made existing systems almost obsolete. Many current operations involving United States forces are international efforts made up of an international coalition. Coalition commanders from different nations must be able to communicate and exchange information. Finally, current fiscal constraints and future DoD budget cuts demand a more efficient and effective method of information exchange between the service components. Duplication of effort and redundancy must also be eliminated while providing DoD connectivity to other Federal agencies like the National Security Agency (NSA) and the Central Intelligence Agency (CIA). The Global Command and Control System (GCCS) will help eliminate many of these problems.

The concept of GCCS is to provide a seamless global information sphere that will support the warrior on the battlefield, today and in the future. Global Command and Control System can be realized by implementing a migratory approach integrating existing stovepipe systems with the gradual introduction of new systems. It will focus on

integrating and reusing as much hardware and software as possible to minimize overall life cycle cost. Below is a list of the eight primary objectives for GCCS [Ref. 38]:

1. Reduce information "push" and replace it with information "pull" as required by the commander. This would eliminate the automatic transmission of information (push) and replace it with selective information retrieval (pull) that can be defined by individual commander's information requirements.

2. Eliminate existing stovepipe systems and provide a single fully integrated system supporting information exchange.

3. Employ modular software developed using open system design to support future growth and software updates at minimum cost.

4. Provide a system capable of handling information at all security levels by implementing a multilevel security system. In the interim a system capable of handling information up to a secret classification will be installed pending the development of multilevel security software.

5. Eliminate duplication of effort in data entry and collection by different federal agencies and the armed forces.

6. Collect and display the information in a user friendly way that will be transparent to the user.

7. Include online easy to use tutorials and help tools.

8. Use commercial-off-the-shelf (COTS) hardware and software whenever possible to minimize cost while ensuring the use of the most current technology available.

GCCS represents a huge undertaking, and it will take several years to complete the entire project. Currently there are several unified and specified commands that have the proof-of-concept system installed.

The system will be implemented using a three step evolutionary development. Beginning with the proof-of-concept system, GCCS will evolve through two major

upgrades termed block one and block two. The proof-of-concept system will include the following initial applications:

1. Navy Operations Support System (OSS).

2. Contingency Theater Automated Planning System (CTAPS).

3. GCCS Status of Resources and Training System (GSORTS).

4. Dynamic Analysis and Replanning Tool (DART).

5. Crisis Management System (CMS).

After successful implementation of the first phase, commanders will select the applications that proved to be the most advantageous. Block one will include those applications and introduce the following applications from the World Wide Military Command and Control System (WWMCCS) technology insertion program (TIP):

1. Force Augmentation Planning and Execution System (FAPES).

2. Logistics Sustainment and Feasibility Estimator (LOGSAFE).

3. Joint Flow and Analysis System for Transportation (JFAST).

Finally, after completion of the block one installation, the block two installation will take place. Block two will add the scheduling and movement (S&M) functionality required for the production of the Time Phased Force Deployment Data (TPFDD). This data is crucial to ensure a smooth transition for force deployment, redeployment and sustainment. [Ref. 39]

The Global Command and Control System will provide a highly mobile, deployable, world wide C^2 system supporting both joint and combined forces spanning the entire spectrum of military operations [Ref. 40]. It is designed to meet the C^2

requirements at the National, theater and Joint Task Force (JTF) levels while including other organizations like the United Nations (UN) and the Federal Emergency Management Agency (FEMA). This three level architecture is illustrated in Figure 34. By providing real time information across an array of functions, commanders will be able to communicate both vertically and horizontally. Real time display of target information and force disposition will provide a fused real time representation of the battlespace. The system will be supported by the Defense Information Systems Network (DISN) and will eventually replace WWMCCS. The system will be .able to support simultaneous operations in different theaters and automatically update changes in force levels.

Open systems architecture will be used to enhance vendor independence and ensure a modular design for simple and inexpensive future system expansion. The current design will include capabilities like multiple video teleconferences, world class graphics, distributed data base and an automated message handling system with reserve bandwidth to support message surges like those experienced during Desert Storm. It will support simultaneous war gaming without interfering with current operations. Near instantaneous transmission of mission essential imagery will be delivered to the respective combat units by employing data compression to conserve bandwidth. [Ref. 40]

A B-ISDN architecture supported by ATM service will provide the backbone for GCCS and support the multimedia communications services mentioned above. Using distributed software and databases, commanders will have near real time access to almost every "bit" of information available. As technology continues to improve and more services become available, expected throughput for GCCS will be approximately 2.265 Mbps using data compression with only a 250 ms delay from the source system to the user [Ref. 41]. Note that most of the 250 ms delay is due to the satellite uplink/downlink and not the pre/postprocessing of data.

94
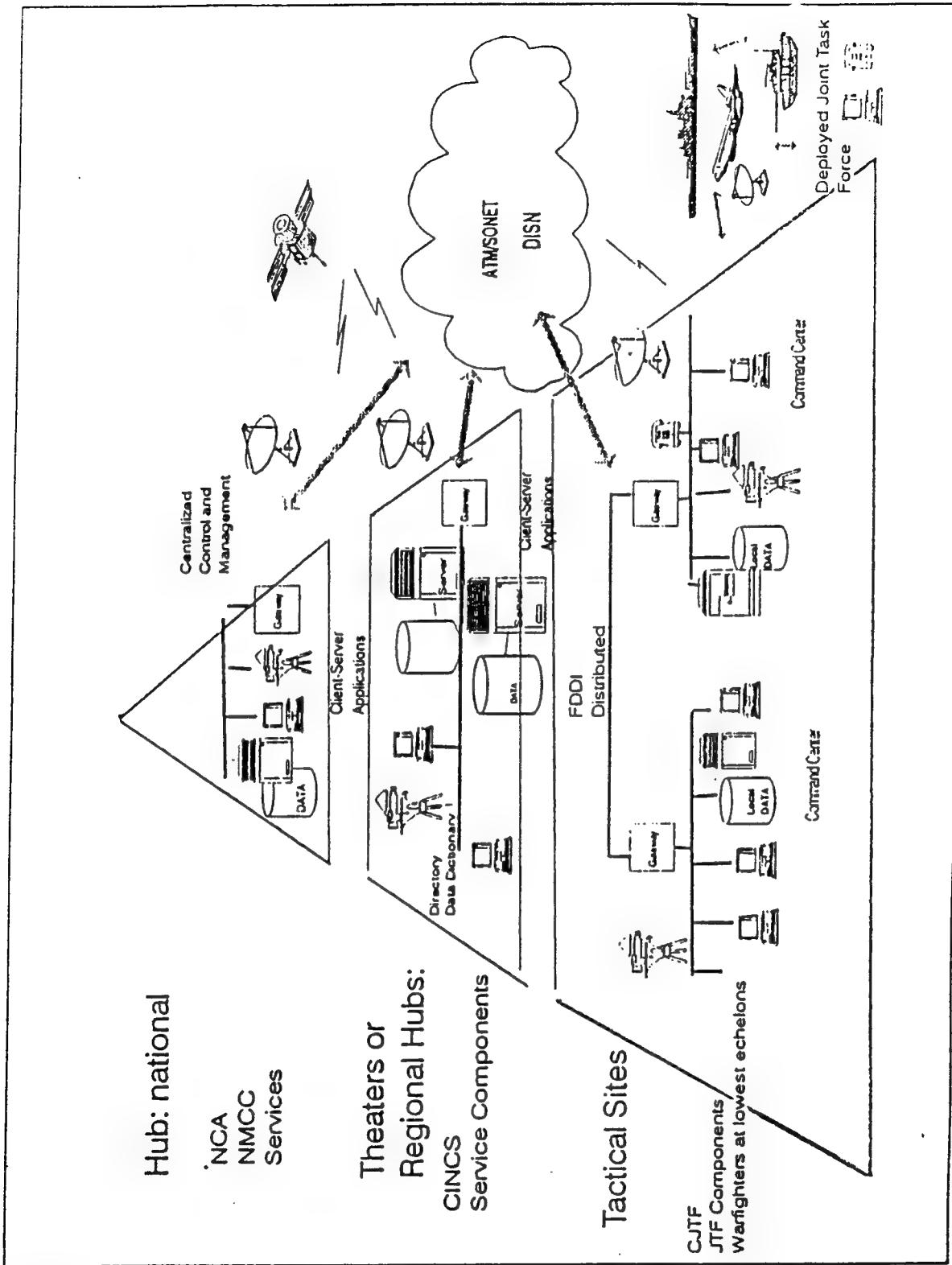
Figure 30. The three levels of GCCS from [Ref. 40].

B. APPLICATION OF STANDARDS TO GCCS

Two approaches are used to discuss the application of data compression standards to GCCS. One is to discuss how data compression standards in general can contribute to the realization of GCCS. Then the application of specific standards to GCCS is discussed.

Data compression can contribute to the achievement of several GCCS objectives. The four GCCS objectives that could be affected by data compression standards are: interoperability and information exchange, modular design, efficient and effective use of assets, and fiscal constraints. Using a common data compression standard for a specific type of data will help ensure interoperability and promote information exchange. A common data compression standard provides a standard data format for transmission as well as standardized encoder and decoder requirements, providing support for increased system interoperability and information exchange. The JITC performs testing and evaluation of new "joint systems" to ensure they are in fact interoperable. Increasing system interoperability and information exchange can also result in more effective and efficient use of existing assets. By providing general encoder and decoder requirements, the standard contributes to open system (modular) design by allowing some degree of design innovation. This freedom in design also supports easier and less costly future upgrades. Employing data compression can significantly reduce bandwidth requirements resulting in more efficient and effective use of available bandwidth. Reducing bandwidth requirements could also lessen the existing dependency on leased commercial satellite channels reducing transmission costs.

Current communications systems are quickly saturated with message traffic, digitized imagery, facsimile and video teleconferencing transmissions. When conducting large operations, like Desert Storm, this can create an operational nightmare. Flash message traffic taking several hours to receive and over eight hours to transmit the air tasking order (ATO) are unacceptable. Sending U.S. military equipment and personnel to other nations' headquarters to enhance communications between coalition forces is

96

unacceptable. International data compression standards like those discussed in previous chapters can contribute to solving these problems.

Standards like JBIG, JPEG and MPEG are recommendations made by the CCITT to the international community. If the United States and other military forces use these commercial standards rather than developing specific military standards, then soon everyone has common encoders and decoders using the same data format. A common data format coupled with compatible encoders and decoders can greatly enhance interoperability and information exchange among systems employed by both coalition and joint forces as well as between the military and other federal agencies. Enhancing interoperability between the military and other federal agencies would also contribute to more effective use of existing national assets. The compression of data prior to transmission would contribute to more efficient use of national assets.

Military employment of international standards, like those discussed in this thesis, support the use of commercial products which in turn contributes to reduced system life cycle costs. Most commercial products produced use international standards providing a global market place for the product in question. Military use of these commercially produced systems could contribute to improved interoperability between coalition forces. Military systems often use technology that is years behind the technology being used by commercial systems. Using commercial systems is one way to get state-of-the-art technology onto the battlefield.

International standards could eliminate the need to develop military standards. All the international standards discussed in this thesis are currently implementable with existing technology and many have already been implemented. True interoperability is much harder to achieve when coalition nations each develop their own specific military standards. Different military standards may use different data formats as well as different encoding and decoding techniques, resulting in the need for data translators. The DoD should begin moving away from the use of military standards and begin to use these international standards if interoperability is to become a reality.

The application of these international standards provides better performance and uses current technology. Voice communications should use MPEG-1 layer-3 to compress the audio signal. Still imagery being transmitted to the commander on the battlefield must occur in real-time and should use the JPEG standard to compress the data. Many commanders desire Cable Network News (CNN) in their operations centers and use VTC when planning joint operations. Using MPEG to compress the video and associated audio signals could provide up to ten times the throughput using satellite communications [Ref. 42]. For example, during operation Desert Storm a great deal of information was exchanged using secure facsimile. Implementation of the JBIG standard could reduce the bandwidth required for secure facsimile by 30 percent.

In summary, GCCS is going to provide a global network for real-time information exchange among commanders. GCCS will encompass all forms of data to include text, still image, facsimile, audio, and video. Existing communications systems cannot support the amount of information exchange invisioned using GCCS. Current communications systems already operate at maximum capacity, and commercial communications channels are now being leased. Data compression can reduce the bandwidth requirements which could reduce the dependency on commercially leased bandwidth. Implementation of international data compression standards by the DoD would serve to support employment of GCCS, enhance combined and joint force interoperability, and reduce the military dependence on commercial communications.

IX. CONCLUSION

There are several problems plaguing todays military. They need to do things smarter, more efficiently and effectively while using fewer resources. Both assets and information must be shared among the armed services and between the military and other federal agencies. There is a growing need to promote interoperability and smooth information exchange at both the joint and combined levels. The use of international standards can help solve many of these problems in an efficient and cost effective manner.

Military standards can promote joint interoperability, but they cannot do much for combined interoperability. International standards can promote both, providing the multi-national forces agree to use them. Most commercial industries use these international standards supporting the use of COTS products. The DoD should promote the use of international standards wherever possible and use NATO as a medium to influence the Europeans and other allies to do the same. Implementing international standards will better serve the military and federal agencies in the long term.

Several military standards employ older technology and should be replaced. The bi-level image compression standard MIL-STD-188-196 should be replaced by the JBIG standard after JBIG has been implemented in commercial products. Using JBIG could provide up to 30 percent more compression over the current MIL-STD that is based on the Group 3 recommendation [Ref. 6]. MIL-STD-188-198A is for the most part a repeat of the JPEG standard, so it is redundant. No military standard exists for video compression although H.261 is considered for VTC applications. Implementing the MPEG-1 standard to reduce the bandwidth requirements for CNN in the joint operations centers could free up bandwidth for other applications. Unmanned aerial vehicles could be used for live video reconnaissance and employ MPEG-1 or perhaps MPEG-4, when available, to minimize bandwidth. MIL-STD-188-113 provides four methods for performing A/D conversion of audio signals. Although some degree of compression can be achieved, only one of these methods— CVSD— actually specifies a compression ratio. A better choice might be the MPEG audio layer-3 providing state of the art technology

99

supporting COTS products while decreasing bandwidth in support of both combined and joint operations. The increase in message traffic during armed conflicts and the resulting saturation of communications channels should indicate the need for a text compression standard. However, no such standard could be found in the commercial or military sectors. Lossless compression of message traffic could essentially reduce the bandwidth required by a factor of at least one-half, assuming a 2:1 compression ratio.

The areas of text, video and audio compression and applications to military communications channels should be explored. Discussion of a standard for text compression is also required to help ensure interoperability. The results provided in Chapter II for the Shannon-Fano and PKZIP compression schemes indicate that Shannon-Fano coding outperforms PKZIP for lossless text compression in noisy channels. These results are not conclusive and further study is required. A more detailed study of the application of international standards to military C^3 systems should be undertaken. This study should focus on items like implementation, performance gain over existing military standards and long term military benefit. More research could also be done on possible application of new technologies, such as wavelets and fractals, and the implementation of new emerging international standards like MPEG-4 or MHEG.

APPENDIX A: ACRONYMS

1D	one dimensional
2D	two dimensional
A/D	Analog-to-Digital
ADPCM	Adaptive Differential Pulse Code Modulation
ANSI	American National Standards Institute
APC	Adaptive Predictive Coding
ARIDPCM	Adaptive Recursive Interpolated Differential Pulse Code
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
BAC	Binary Asymmetric Channel
B-ISDN	Broadband ISDN
BSC	Binary Symmetric Channel
bpp	bits per pixel
C^2	Command and Control
C^3	Command, Control and Communications
CATV	Cable Television
CCIR	Consultive Committee on International Radiocommunications
CCITT	Consultive Committee on International Telegraphy & Telephony
CIA	Central Intelligence Agency
CIF	Common intermediate format
CMS	Crisis Management System
codec	encoder/decoder
COTS	Commercial-off-the-shelf
CPB	Constrained Parameter Bitstream
CR	Compression Ratio
CTAPS	Contingency Theater Automated Planning System
CVSD	Continuously Variable Slope Delta Modulation
D/A	Digital-to-Analog

DART	Dynamic Analysis and Replanning Tool
DCT	Discrete Cosine Transform
DISN	Defense Information Systems Network
DoD	Department of Defense
DPCM	Differential Pulse Code Modulation
EOB	End-of-Block, or end-of-band (progressive JPEG)
FAPES	Force Augmentation Planning and Execution System
FAQ	Frequently Asked Questions
FDCT	Forward Discrete Cosine Transform
FEC	Forward error correction
FEMA	Federal Emergency Management Agency
FFT	Fast Fourier Transform
FIPS	Federal Information Processing Standard
FIR	Finite Impulse Response
GCCS	Global Command and Control System
GSORTS	GCCS Status of Resources and Training System
HDTV	High Definition Television
Hz	Hertz
IDCT	Inverse Discrete Cosine Transform
IFS	Iterated Function System
ISDN	Integrated Services Digital Network
ITU	International Telecommunications Union
ITU-T	International Telecommunications Union and Consultive Comittee on International Telegraphy & Telephony
JBIG	Joint Bi-Level Imagery Experts Group
JFAST	Joint Flow and Analysis System for Transportation
JITC	Joint Interoperability Test Center
JPEG	Joint Photographic Experts Group
JTF	Joint Task Force

Kbits	Kilo bits
kbps	kilo bits per second
kHz	kilo-hertz
LOGSAFE	Logistics Sustainment and Feasibility Estimator
LPC	Linear Predictive Coding
LSB	Least Significant Bit
LZ	Lempel-Ziv coding
LZW	Lempel-Ziv-Welch coding
MB	Macro block
Mbps	Mega bits per second
MDCT	Modified DCT
MHEG	Multimedia Hypermedia Experts Group
MIL-STD	Military Standard
MPEG	Moving Pictures Expert Group
ms	millisecond
MSB	Most Significant Bit
NITF	National Imagery Transmission Format
NITFS	National Imagery Transmission Format Standard
NSA	National Security Agency
OSS	Navy Operations Support System
PC	Prediction Coefficient
PCM	Pulse Code Modulation
PIFS	Partial Iterated Function System
pixel	picture element
QCIF	Quarter-CIF
RC	Reflection Coefficient
RGB	Red-Green-Blue color space
rms	root mean square
S&M	Scheduling & Movement

SIDS	Secondary Imagery Dissemination System
SIF	Source Input Format
SMR	Signal-to-Masking Ratio
SNR	Signal-to-noise ratio
TIP	Technology Insertion Program
TPFDD	Time Phased Force Deployment Data
UN	United Nations
VCC	Virtual Channel Connection
VLSI	Very large scale integration
VPC	Virtual Path Connection
VTC	Video Teleconferencing
VTP	Video Telephony
WWMCCS	World Wide Military Command & Control System
YC_bC_r	Luminance-chrominance color space

APPENDIX B: STANDARDS BODIES AND POINTS OF CONTACT

This appendix provides a list of locatiions for obtaining copies of standards. It also provides a list of experts, working within the Department of Defense, in the field of data compression.

A. ADDRESSES OF STANDARDS BODIES

1. Federal Information Processing Stanards (FIPS)

   a. DoD Activities

   Commanding Officer
   Naval Publications and Forms Center
   (ATTN: NPODS)
   5801 Tabor Avenue
   Philadelphia, PA 19120-5099

   b. Others

   National Technical Information Service (NTIS)
   5285 Port Royal Road
   Springfield, VA 22161-2171
   Telephone: 703-487-4650

2. Federal and Military Standards and Specifications

   a. DoD Activities

   Commanding Officer
   Naval Publications and Forms Center
   (ATTN: NPODS)
   5801 Tabor Avenue
   Philadelphia, PA 19120-5099

   b. Others

   General Services Administration
   GSA Specification Unit (WFSIS)
   Room 6039
   7th & D Streets SW
   Washington, DC 20407
   Telephone: 202-472-2205

3. ITU-T Recommendations

    a. International Telecommunications Union
       Place Des Nations
       CH-1211
       Geneva 20, Switzerland
       Telephone: +41 22 730 5111
       Fax: +41 22 733 7256
       E-Mail: helpdesk@itu.ch

    b. Omnicom, Phillips Business Information Inc.
       1201 Seven Locks Road
       Suite 300
       Potomac, MD 20854
       Telephone: 1-800-666-4266
       Fax: 1-301-309-3847

    c. Document Center
       1504 Industry Way (Unit 9)
       Belmont, CA 94002
       Telephone: 415-591-7600
       Fax: 415-591-7617

4. ITU/CCITT Recommendations

    a. American National Standards Institute (ANSI)
       11 W. 42nd Street
       New York, NY 10036

5. Electronic Industry Association (EIA) Publications

    a. Electronic Industry Association Standards Sales
       2001 Pennsylvania Avenue NW
       Washington, DC 20006
       Telephone: 202-457-4966
       Fax: 202-457-4985

B. DATA COMPRESSION CONTACT PERSONS IN DoD

Below is a listing of data compression experts that were contacted at some point during the research phase. All the phone numbers provided are DDN unless preceded by a three digit area code.

1. Defense Information Systems Agency (DISA)

    a. Section Chief: Mr. Ed Kovanic (992-7715)

b. Still Image Compression: Mr. Manny Valstakis (992-7715)

c. Video Compression: Mr. Claus Rittenbach (992-7715)

d. Audio Compression: Mr. Ralph Ligaouri (992-7715)

2. DISA Center for Standards

a. Point of Contact: Ms. Carol Ciepiela (703-487-3536)

3. Joint Interoperability Test Center (JITC)

a. NITFS: Mr. William Durham (879-5458)

APPENDIX C: MATLAB CODE

This appendix provides the MATLAB code used in the analysis reported in Chapter II.

A. SHANNON-FANO ALGORITHM IN MATLAB

FILE: SF.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
This program was written by Scott Pratt to perform Shannon-Fano coding on a    %
given text file.  The text file is read in in ASCII format and then compressed         %
using the Shannon-Fano algorithm.  This program also includes a binary                 %
symmetric channel to simulate transmission of the compressed data.  After receipt   %
ot the transmitted data the received file is expanded using the Shannon-Fano          %
decoder.  Finally, a comparrison is performed between the original ASCII file and  %
the decoded file to check for errors.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% READ IN THE INPUT FILE

fid=fopen ('b:samp1.wpd','r');  % NOTE: you must change the name of the file
'samp.wpd'.
A=fread(fid,inf,'char');
total=length(A);   % determines the number of characters in the input file.

% This section determines the frequency of occurrence of each unique character (As)
% in the input file and associates the frequency with the respective character (X).

nb=input('Input the number of bins (128 or 256) ');
Z=[1:1:nb];
[As,X]=hist(A,Z);

% This section determines the probability of occurrence of each character (p) and
%  then calculates the integer length (l) of the respective codeword.

p=zeros(1,nb);   % initialize probability vector
l=zeros(1,nb);   % initialize length vector
for i=1:nb
    p(i)=As(i)/total;     % computes the probability of occurrence of the character
    if p(i)==0
        l(i)=inf;
    else
```

```
            l(i)=round(log(1/p(i))/log(2));     % computes the length of the respective
 end                                            % codeword
end
Lm=min(l);
LM=0;
for i=1:nb
    if l(i)~=inf & l(i)>LM
          LM=l(i);     % determines max(l)
    end
end


% This section assigns the codewords (in decimal) based upon their respective length
% where the shortest length has a smaller codeword and the longest length has a
%  larger codeword.

disp('Commence codeword determination process')
d=(LM-Lm)+1;
c=1;
L=Lm;
cw=zeros(1,nb);     % initialize the codeword vector
for a=1:d
    for i=1:nb
        if l(i)==L
            cw(i)=c;     % assigning the codewords (in decimal)
            c=c+1;
        end
    end
    L=L+1;
end


% This section encodes the original file (A) and places the encoded data into the
% codeword file (CW).  Note that the encoded data is still in decimal format.

disp('Commence encoding process')
CW=zeros(1,total);     % initialize the encoded vector
for a=1:total
    for i=1:nb
        if A(a)==X(i)
            CW(a)=cw(i);     % encoding process
        end
    end
end
```

```
% This section converts the decimal codewords in CW to variable length binary
% codewords (Cwb) one symbol at a time.  It then  simulates a binary symmetric
% channel with the P(1|1) and P(0|0) provided as inputs to allow for simulating a
% binary symmetric channel with different noise levels.  The file being transmitted
% is the binary codeword file (Cwb) and the received file (Cwbr) will be the one
% that is binary-to-decimal converted and decoded.

Cwdr=size(CW);
P1=input('Input the P(1|1) ');    % prob. of receiving a 1 given a 1 was sent
P0=input('Input the P(0|0) ');    % prob. of receiving a 0 given a 0 was sent
ct=0;
cti=0;
out=[];
for a=1:total
    Cwb=d2b(CW(a));        % decimal-to-binary conversion process
    ct=ct+length(Cwb);
    out=[out Cwb];

% begin simulated binary symmetric channel

    Cwbr=[];
    for i=1:length(Cwb)
        k=rand;
        if Cwb(i)==1
            if k <= P1
                Cwbr(i)=1;
            else
                Cwbr(i)=0;
            end
        else
            if k <= P0
                Cwbr(i)=0;
            else
                Cwbr(i)=1;
            end
        end
    end
    for i=1:length(Cwb)
        if Cwbr(i)~=Cwb(i)
            cti=cti+1;            % counts the number of bit errors received
        end
    end
    Cwdr(a)=b2d(Cwbr);      % binary-to-decimal conversion process
```

```
end

% This section decodes the encoded data after receipt (Cwdr) and places the
% respective ASCII character into the resultant file (Ar).

disp('Commence decoding')
Ar=zeros(1,total);     % initialize the receipt vector
for a=1:total
    for i=1:nb
        if Cwdr(a)==cw(i)
            Ar(a)=X(i);     % decoding process
        end
    end
end

% This section compares the decoded file (Ar) with the original file (A) and
% This section computes the number of errors (E) in the decoded receipt file (Ar).

E=0;
for a=1:total
    if Ar(a)~=A(a)
        E=E+1;             % computes the number of errors in the decoded file
    end
end
E
Eb=cti
bits=length(out)
ct

% This section computes the compression ratio (CR) achieved using
% this algorithm.

CR=(total*8)/ct

% This section writes Ar to a floppy disk in drive B.  Note the name of
% the file on the disk must be specified where noted below.  The default file
% name is 'rcv'.


fclose(fid);
fidout=fopen('b:rcv','w');     %NOTE:  Specify filename here in place of 'rcv'.
count=fprintf(fidout,'%s \n',Ar);
fclose(fidout);
```

FILE: B2D.M
function y=b2d(b)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This program was written by Scott Pratt to perform binary to decimal
%  conversion for variable length binary numbers.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

d=0;
b=fliplr(b);
for i=1:length(b)
   d=((2^(i-1)).*b(i))+d;        % binary to decimal conversion process
end
y=d;
```

FILE:D2B.M
function y=d2b(d)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This program was written by Scott Pratt to perform decimal to binary
%  conversion resulting in variable length binary numbers.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a=0;
if d < 2
   a=1;
elseif 1 < d & d < 4
   a=2;
elseif 3 < d & d < 8
   a=3;
elseif 7 < d & d < 16
   a=4;
elseif 15 < d & d < 32
   a=5;
elseif 31 < d & d < 64
   a=6;
elseif 63 < d & d < 128
   a=7;
else
   a=8;
end
for i=1:a                    % commence binary to decimal conversion
   b(i)=rem(d,2);
   d=floor(d/2);
```

```
    if d<1
        d=0;
    end
end
b=fliplr(b);
y=b;
```

B. BINARY SYMMETRIC CHANNEL MATLAB CODE

FILE:BCH.M
%%%
% This program was written by Scott Pratt to simulate a binary symmetric
% channel using three inputs. The first input is the probability of
% successfully receiving a 1 given a 1 was sent P(1|1). The second input is
% the probability of successfully receiving a 0 given a 0 was sent P(0|0).
% The last input is reading in the ASCII data file to be transmitted. The
% file name must be changed below where noted and the resultant output
% file is written to drive B with file name 'file2'.
%%%

```
P1=input('Input P(1|1) ');  % probability of successfully receiving a 1 given a 1 was
                            % sent
P0=input('Input P(0|0) ');  % probability of successfully receiving a 0 given a 0 was
                            % sent
Results=[];
fid=fopen('b:samp3.wpd','r');       % NOTE: enter file name here in place of
                                    % 'samp.wpd'
A=fread(fid,inf,'char');
disp('Commence decimal-to-binary conversion')
Sent=zeros(length(A),8);    % initialize sending matrix
for a=1:length(A)
    Sent(a,:)=dec2bin(A(a));     % decimal-to-binary conversion
end
Rec=size(Sent);
disp('Start Binary Channel')
for a=1:length(A)
    for i=1:8                   % routine to simulate binary channel
        k=rand;
        if Sent(a,i)==1
            if k<=P1
                Rec(a,i)=1;
            else
                Rec(a,i)=0;
```

114

```
                end
            else
                if k<=P0
                    Rec(a,i)=0;
                else
                    Rec(a,i)=1;
                end
            end
        end
    end
disp('Commence binary-to-decimal conversion')
for a=1:length(A)
    R(a)=bin2dec(Rec(a,:));      % binary-to-decimal conversion
end
X=[1:1:128];
for a=1:length(R)
    for i=1:length(X)
        if R(a)==X(i)
            R(a)=X(i);
        end
    end
end
fclose(fid);
fidout=fopen('b:file2','w');
charout=fprintf(fidout,'%s \n',R);
fclose(fidout);

% compute the number of errors in the received
%  file (R)

ct=0;
for a=1:length(A)
    if R(a)~=A(a)
        ct=ct+1;
    end
end
Error=ct
cti=0;
for a=1:length(A)
    for i=1:8
        if Sent(a,i)~=Rec(a,i)
            cti=cti+1;
        end
```

```
        end
end
Ebit=cti
```

The following MATLAB files were written by Dr. Paul Moose.

```
DEC2BIN.M   performs decimal to binary conversion
BIN2DEC.M   performs binary to decimal conversion
SCOTT.M      (modified by the author) Channell Two simulation
AWGN.M       AWGN generator
CNCD.M       convolutional coder
EUCDIS.M    computes a euclidean distance
PSK.M         simulates M-PSK modulation (specify M)
CHECK.M     counts the number of errors in the received file
FEC.MAT      generates the two matrices G and T used for FEC
DMOD.M      .QPSK demodulator
SCOTTX..M   Channel Three simulation
```

LIST OF REFERENCES

1. Hamming, Richard W., *Coding and Information Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

2. "National Imagery Transmission Format," MIL-STD-2500, 18 June 1993.

3. Gersho, Allen and Gray, Robert M., *Vector Quantization and Signal Compression*, pp. 277-295, Kluwer Academic Publishers, Boston, MA, 1992.

4. PKWARE, "APPNOTE.TXT", PKWARE Bulletin Board System, (414)354-8670, 1994.

5. Lin, Shu and Costello Jr., Daniel J., *Error Control and Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

6. Gailly, Jean-loup, "Introduction to JBIG," comp.compression Frequently Asked Questions, #74, rtfm.mit.edu, /pub/usenet/news.answers/compression-faq, 16 June 1994.

7. Gailly, Jean-loup, "What is the state of the art in lossless image compression?," comp.compression Frequently Asked Questions, #16, rtfm.mit.edu, /pub/usenet/news.answers/compression-faq, 16 June 1994.

8. Pennebaker, William B. and Mitchell, Joan L., *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, NY, 1993.

9. Pennebaker, W.B., Mitchell, J.L., Langdon Jr., G.G. and Arps, R.B., "An Overview of the Basic Principles of the Q-coder Adaptive Binary Arithmetic Coder," IBM Journal of Research and Development: Q-Coder, vol. 32, no. 6, pp.717-726, November 1988.

10. "Bi-Level Image Compression," MIL-STD-188-196, 18 June 1993.

11. Gailly, Jean-loup, "Introduction to JPEG," comp.compression Frequently Asked Questions, #75, rtfm.mit.edu, /pub/usenet/news.answers/compression-faq, 16 June 1994.

12. Wallace, Gregory K., "The JPEG Still Picture Compression Standard," Communication of the ACM, vol.34, no.4, pp.31-44, April 1991.

13. Schremp, Doug and Weiss, Jeffery, " Putting Data on a Diet," IEEE Spectrum, pp. 36-39, August 1993.

14. Durham, William, personal communication, JITC, DDN: 879-5458, April, 1994.

15. "Adaptive Recursive Interpolated Differential Pulse Code Modulation (ARIDPCM) Compression Algorithm," MIL-STD-188-197, 18 June 1993.

117

16. "Joint Photographic Experts Group (JPEG) Image Compression," MIL-STD-188-198A, 15 December 1993.

17. Liou, Ming, "Overview of the p×64 kbits/s Video Coding Standard," Communications of the ACM, vol.34, no.4, pp.60-63, April 1991.

18. Le Gall, Didier, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, vol.34, no.4, pp.46-58, April 1991.

19. Fogg, Chad, "MPEG-2," The MPEG-FAQ, Version 3.1, phade@cs.tu-berlin.de, 14 May 1994.

20. Popp, Harold, "Introduction to MPEG," comp.compression Frequently Asked Questions, rtfm.mit.edu, /pub/usenet/news.answers/ compression-faq, March 1994.

21. Fogg, Chad, "MPEG Frequently Asked Questions List," Draft 3.4, cfogg@netcom.com, 18 June 1994.

22. "MPEG-2 Encoder/Decoder," Version 1.1, MPEG-L@netcom.com, June 1994.

23. "Interoperability and Performance Standard for Video Teleconferencing," MIL-STD-188-331, 29 March 1994.

24. "Video Teleconferencing Services at 56 to 1920 kb/s," Federal Information Processing Standard Publication 178, 21 December 1992.

25. Popp, Harold, "INFO.TXT for MPEG Audio Layer-3 Shareware Code," version 1.47, layer3@iis.fhg.de, 08 June 1994.

26. Brandenburg, Karlheinz and Stoll, Gerhard, "The ISO/MPEG-Audio Codec: A Generic Standard for Coding of High Quality Digital Audio," Audio Engineering Society Convention, 1994.

27. Popp, Harold, "ISO-MPEG Audio Layer-III," 27 January 1994.

28. Grill,B., Herre, J., Brandenburg, K.H., Eberlin, E., Koller, J. and Muller, J., "Improved MPEG-2 Audio Multi-Channel Encoding," Audio Engineering Society Convention, 1994.

29. "Interoperability and Performance Standards for Analog-to-Digital Conversion Techniques," MIL-STD-188-113, 17 February 1987.

30. "Analog-to-Digital Conversion of Voice by 2400 bit/second Linear Predictive Coding," Federal Standard 1015, 24 November 1984.

31. Argent, Fabrizio, Benelli, Giuliano and Kutufa, Claudio, "Transmission of Wavelet Transform Coded TV Images on a Noisy Channel," IEEE International Conference on Communications, Geneva, vol.1, pp. 386-389, 1993.

32. Kirk, Richard, "What is Wavelet Theory," comp.compression Frequently Asked Questions, rtfm.mit.edu, /pub/usenet/news.answers/ compression-faq, 16 June 1994.

33. Reusens, Emmanuel and Ebraham, Tourady, "New Results in Subband/Wavelet Image Coding," IEEE International Conference on Communications, Geneva, vol.1, pp. 381-385, 1993.

34. Kominek, John, "Introduction to Fractal Compression," comp.compression Frequently Asked Questions, rtfm.mit.edu, /pub/ usenet/news.answers/compression-faq, 16 June 1993.

35. Stallings, William, *Data and Computer Communications*, pp. 819-830, Macmillan Publishing Company, New York, NY, 1994.

36. Stallings, William, *Data and Computer Communications*, pp.791-796, Macmillan Publishing Company, New York, NY, 1994.

37. Colaitis, Francoise and Leger, Alain, "MHEG: An Emerging Standard for the Interchange of Multimedia and Hypermedia Objects," IEEE International Conference on Communications, Geneva, vol.1, pp. 542-546, 1993.

38. Edmonds, Albert J., Global Command and Control System: The Warriors Tool, Joint Staff (J6), Government Publication, 1993.

39. Edmonds, Albert J. (Ed.), Global Command and Control System Newsletter, no.1, 15 December 1993.

40. Global Command and Control System Concept of Operations, Draft, Joint Staff (J6), Government Publication, 31 January 1994.

41. Jo, Kenneth Y., "Target GCCS Architecture Modeling and Simulation," Briefing at NPS, March 1994.

42. Laczko, Frank, "The Motion Pictures Expert Group Digital Compression Standard and its Role in Satellite Systems," IEEE International Conference on Communications, New Orleans, LA, pp.117-119, 1994.

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Technical Information Center 2
   Cameron Station
   Alexandria VA 22304-6145

2. Library, Code 052 2
   Naval Postgraduate School
   Monterey CA 93943-5002

3. Code EC 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121

4. Prof. Paul Moose, Code EC/Me 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121

5. Prof. Murali Tummala, Code EC/Tu 5
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121

6. Dr. Robert Fallon 1
   J6E
   Joint Staff
   Pentagon
   Washington D.C. 20318-9300

7. Deborah R. Castleman 1
   Deputy Assistant Secretary of Defense
   Command, Control & Communications
   Pentagon, Rm 3E194
   Washington D.C. 20318-3040

8.   LT Everett S. Pratt   1
        1 Ayers Ln
        Dover, New Hampshire 03820